НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.43

«До захисту допущено»

Завідувач кафедри
_____ І.Р. Пархомей
(підпис)

“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 126 «Інформаційні системи та технології»

на тему: Персональний кабінет студента курсів навчання робототехніці___
_____

Виконав: студент другого курсу, групи ІК-82мп
(шифр групи)

_____Новодранов Артур Сергійович_____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Крилов Є.В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)

«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Новодранову Артуру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Персональний кабінет студента курсів навчання робототехніці»,
науковий керівник дисертації доцент, к.т.н., доцент Крилов Є.В. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 28 » жовтня 2019 р. № 3770-с

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження *процес роботи персонального кабінету студента курсів навчання робототехніці*

4. Предмет дослідження *технології оптимізації швидкості роботи веб-системи*

5. Перелік завдань, які потрібно розробити *аналіз аналогічних існуючих систем; вибір технологій для проектування та розробки веб-системи; розробка персонального кабінету; тестування та збір даних по роботі додатку; оптимізація швидкості роботи персонального кабінету студента* _____
_____

6. Орієнтовний перелік ілюстративного матеріалу – 6 плакатів

7. Орієнтовний перелік публікацій 1. Новодранов А.С. Порівняння функціональних можливостей персональних кабінетів навчальних закладів // Materials of the XV International Scientific and Practical Conference, April 30 – May 7, 2019. 2. Новодранов А.С., Крилов Є.В., Анікін В.К. Оптимізація швидкодії високонавантажених веб-систем // Materials VII International Scientific Practical Conference 07 – 15 November, 2019.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
НК	Пасько В.П., доцент		
Перевірка на співпадіння	Лісовиченко О.І., доцент		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області та постановка задачі	03.09.2019р.	
2	Вибір технологій для розробки персонального кабінету	10.09.2019р.	
3	Проектування бази даних	17.09.2019р.	
4	Розробка основних модулів веб-системи	28.09.2019р.	
5	Оптимізація роботи веб-додатку	15.10.2019р.	
6	Тестування розробленої системи	24.10.2019р.	
7	Оформлення пояснювальної записки	01.11.2019р.	
8	Висновки	15.11.2019р.	

Студент _____ Новодранов А.С.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Крилов Є.В.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У даній роботі розглянуто вирішення проблеми оптимізації веб-додатків.

Насамперед проаналізовано аналогічні системи персональних кабінетів, визначено їх переваги та недоліки. В результаті отриманих даних визначено необхідні модулі системи, необхідні технології для розробки та методи для подальшої оптимізації.

Результатом виконання даної магістерської дисертації є створений персональний кабінет студента курсів навчання робототехніці. Здійснено його оптимізація, за критерій оптимізації якої обрано швидкість завантаження сторінок. Загалом оптимізація персонального кабінету складалась із оптимізації серверної частини системи, оптимізації бази даних, а саме оптимізація складних запитів, об'єднання простих запитів у складні та кешування індексів. Також виконана оптимізація клієнтської частини персонального кабінету. Застосовано систему кешування даних Redis, а також інтегровано масштабування веб-серверу за допомогою технології Amazon EC2 для підтримання роботи системи при високих та пікових навантаженнях.

Ключові слова: оптимізація швидкості роботи персонального кабінету, система навчання студентів, персональний кабінет студента, JavaScript, PHP, MySQL, Amazon EC2, Laravel.

Розмір пояснювальної записки – 97 аркушів, яка містить 36 ілюстрацій, 25 таблиць та 2 додатки.

ABSTRACT

This paper addresses the solution to the problem of web application optimization.

First of all, similar systems of personal cabinets were analyzed, their advantages and disadvantages were determined. As a result of the obtained data, the necessary modules of the system, the necessary technologies for development and methods for further optimization were identified.

The result of this master's thesis is the creation of a personal cabinet of student robotics training courses. It has been optimized and the page load rate has been selected as the optimization criterion. In general, optimization of the personal cabinet consisted of optimization of the server part of the system, optimization of the database, namely optimization of complex queries, combining simple queries into complex queries and caching of indexes. The optimization of the client part of the personal cabinet was also carried out. A Redis data caching system was implemented, as well as scaling the web server with Amazon EC2 technology to keep the system running at high and peak loads.

Keywords: personal office speed optimization, student learning system, student personal office, JavaScript, PHP, MySQL, Amazon EC2, Laravel.

Explanatory note size - 97 pages, contain 36 illustrations, 25 tables and 2 applications.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дисертації

на тему: *Персональний кабінет студента курсів навчання робототехніці*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

PHP – hypertext preprocessor

REST – representational state transfer

API – application programming interface

HTTP – hypertext transfer protocol

MVC – model-view-controller

JSON – JavaScript object notation

PSR – PHP standards recommendations

TDD – test-driven development

СУБД – система управління базою даних

ВСТУП

В наш час інформаційні технології розвиваються дуже швидкими темпами і все більше інтегруються в наше життя. Однією із таких інформаційних систем є безпосередньо мережа інтернет, швидкість розвитку якого займає одне з передових місць, а вплив його на наше життя складно недооцінити. Таке широке розповсюдження та швидкий розвиток він набув в першу чергу через те, що містить велику кількість інформації з різних галузей, а також швидкого та зручного доступу для неї з майже з будь-якого місця на планеті. Будь-який користувач мережі інтернету може отримати необхідну інформацію на свій пошуковий запит. Результат пошукового запиту буде містити велику кількість інформації зібраної із безлічі інтернет ресурсі, яка буде рангована за своєю актуальністю, новизною та унікальністю. Також користувачі можуть створювати або корегувати уже існуючі статі або інші матеріали. Залишилися позаду ті часи коли неможливо розмовляти по стаціонарному телефону та одночасно користуватися мережею інтернет. На даний час доступ до мережі інтернет користувач має майже в будь-якому куточку світу з будь-якого пристрою, такого як комп’ютер, телефон тощо. Завдяки такому широкому та швидкому його розповсюдженню люди можуть переглядати відео у високій якості, мати доступ до світових баз знань тощо. Необхідно зробити наголос на тому, що користувачі не обмежуються ресурсами розміщеними в тій країні де він фізично знаходиться, а має доступ майже до будь-якого ресурсу. Обмеження на доступ до деяких ресурсів можуть накладатися державою або інтернет провайдером із різних причин.

Інтернет надає користувачу необмежені можливості майже в усіх сферах життя. Однією із основних сфер життя є навчання. Саме на навчання кожен з нас у певний життєвий проміжок витратив велику кількість часу. В наш час існує досить багато платформ або додатків як іноземного так і національного виробництва для спрощення процесу навчання у певній галузі яку обирає користувач. Зазвичай такі додатки мають досить широкий список навчальних курсів, серед яких користувач

може обрати один або декілька тих які йому необхідні. Процес навчання на певному курсі складається як правило з відео матеріалів або семінарів в режимі реального часу, теоретичної частини та практичних домашніх завдань. Вся програму курсу поділена на структурні блоки, по завершенню кожного студент зобов'язаний пройти контрольний тест, який складається з теоретичного та практичного матеріалу. Безпосередньо на таких курсах присвячених програмуванню студенти по закінченню всього курсу проходять глобальний тест по всьому матеріалу курсу та здають контрольне практичне завдання, яке є майже повноцінним додатком для включення його в своє портфоліо. По завершенню таких курсів студент отримує сертифікат проходження відповідного курсу. Перевагою такої методики навчання є те, що студент сам обирає що він хоче вивчати, гнучкий графік навчання та підтримку навчального закладу при влаштуванні на роботу. Навчання ж в українських вузах суттєво відрізняється. Студент вступаючи на перший курс обраного Вузу обирає спеціальність для навчання. Далі студент вивчає досить багато суміжних дисциплін, актуальність та необхідність яких знаходиться під питанням. Контроль за успішністю навчання здійснюється раз у семестр у вигляді екзаменів та заліків. В процесі навчання студент визначається з тим напрямком діяльності в якій йому подобається та хотілося б працювати. Він починає самостійно засвоювати матеріал із цієї області, а часу та наснаги на вивчення дисциплін запропонованих вузом не вистачає. На мою думку, це можна вирішити таким чином, щоб студент на певному етапі навчання міг обрати цікаві для нього дисципліни та додати до них перелік обов'язкових.

В даній магістерській дисертації поставлено за мету оптимізувати та підвищити швидкодію та продуктивність персонального кабінету студента навчання робототехніці.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Об’єкт та предмет дослідження

Дослідження проводиться зокрема в області розробки та оптимізації веб-додатків. В наш час існує велика кількість додатків для навчання студентів, але значна їх частина має велику кількість недоліків, до яких можна віднести неналежну оптимізацію, перенасиченість користувацького інтерфейсу, малу швидкодію тощо. Постає потреба в створенні системи, яка б не мала в собі тих недоліків які є у її прямих аналогів. Тому, що така система навчання студентів зазвичай розміщує велику кількість відеоматеріалу, текстового матеріалу та інших модулів, тому питання швидкодії повинно бути на першому місці при розробці. Також необхідно приділити увагу оптимізації цієї система тому, що при завантаженні певної сторінки системи завантажується досить велика кількість інформації, такої як картинки, текст, відео файли тощо, тому необхідно максимально зменшити час завантаження сторінки для зручного та комфортного користування системою. В свою чергу не слід забувати також про навантаження на сервер, через можливу одночасну велику кількість користувачів системою. В цьому випадку на сервер буде здійснюватися навантаження через велику кількість до різноманітних ресурсів сервера так і до бази даних. Тому також слід приділити увагу також і оптимізації бази даних, в саме до скорочення часу відповіді на запит та на довгі запити. На завершення необхідно належним чином протестувати систему, як її серверну частину так і користувацький інтерфейс.

Об’єкт дослідження – процес роботи персонального кабінету студента курсів навчання робототехніці та час виконання запитів користувачів.

Предмет дослідження – технології та методи для оптимізації веб-додатків, для забезпечення його належної роботи.

1.2 Аналіз існуючих систем

В наш час існує безліч додатків для навчання студентів, але більша частина з них мають недоліки, такі як досить великий час завантаження сторінки, застарілий користувацький інтерфейс або перенасичення сторінок зайвими інформаційними блоками. Всі ці фактори призводять до того, що користувачу буде не зручно користуватися цим додатком, довго чекати завантаження сторінок або важко сконцентруватися через велику кількість інформаційних блоків на сторінці. Все це може «відштовхнути» користувача від користування даною системою.

Однією з найбільш відомих систем для навчання студентів є GeekBrains, персональний кабінет якої зображений на рис. 1.1

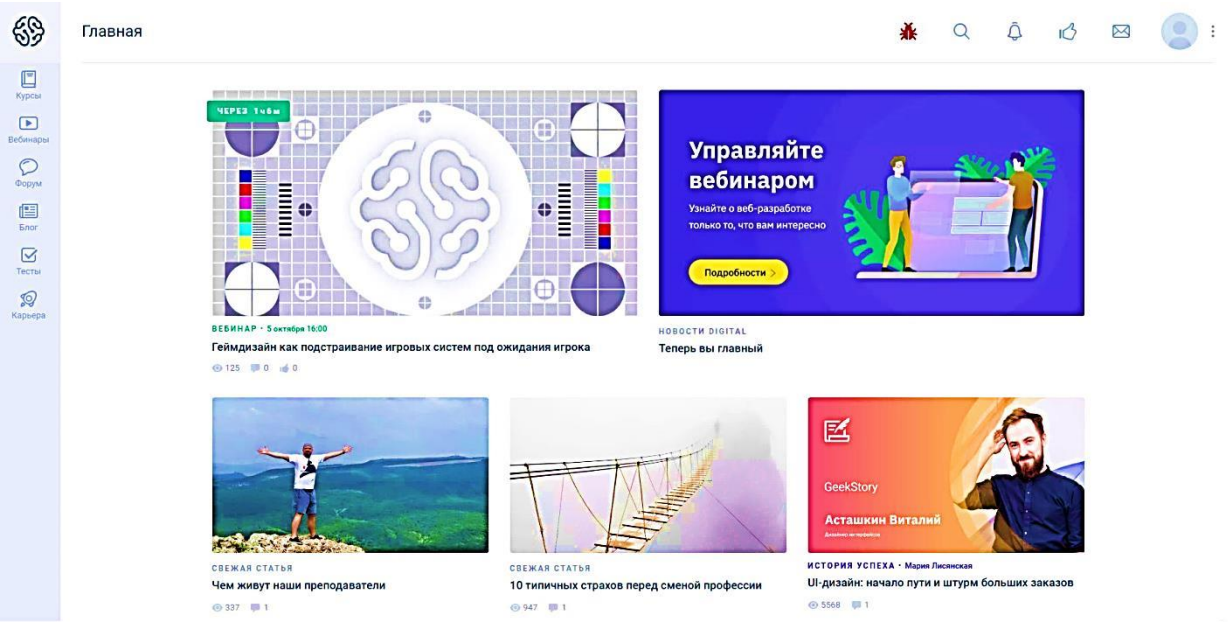


Рисунок 1.1 – Персональний кабінет системи GeekBrains

Дана система для навчання має сучасний, приємній та не загромаджений інформаційними блоками дизайн. Головна сторінка персонального кабінету містить блоки з актуальними новинами про хід навчання, нові курси, викладачів та розважальний контент. Головна сторінка персонального кабінету студента містить бокове меню яке включає в себе: курси – відображення навчальних курсів які обрав для себе студент, вебінари – для перегляду відеоматеріалів в режимі

реального часу, форум – для забезпечення спілкування між студентами певних курсів, для вирішення можливих проблем або запитань, блог – для перегляду цікавих та пізнавальних інформаційних матеріалів від викладачів даного навчального закладу, тести – для проходження контрольних тестів по закінченим блокам обраного курсу, кар'єра – для перегляду можливих оголошень про пошук роботи після закінчення певного курсу. Також головна сторінка містить блок з нагадуванням про найближчі вебінари або інші заходи. У верхній частині кабінету розміщено меню з пунктами особистих повідомлень, нагадувань, пошуку необхідних курсів та фотографія користувача, яка в свою чергу виконана у виглядів кнопки входу до редагування персональної інформації.

Сторінка навчальних курсів виконана в досить стриманому, чистому та сучасному дизайні, та зображена на рис. 1.2

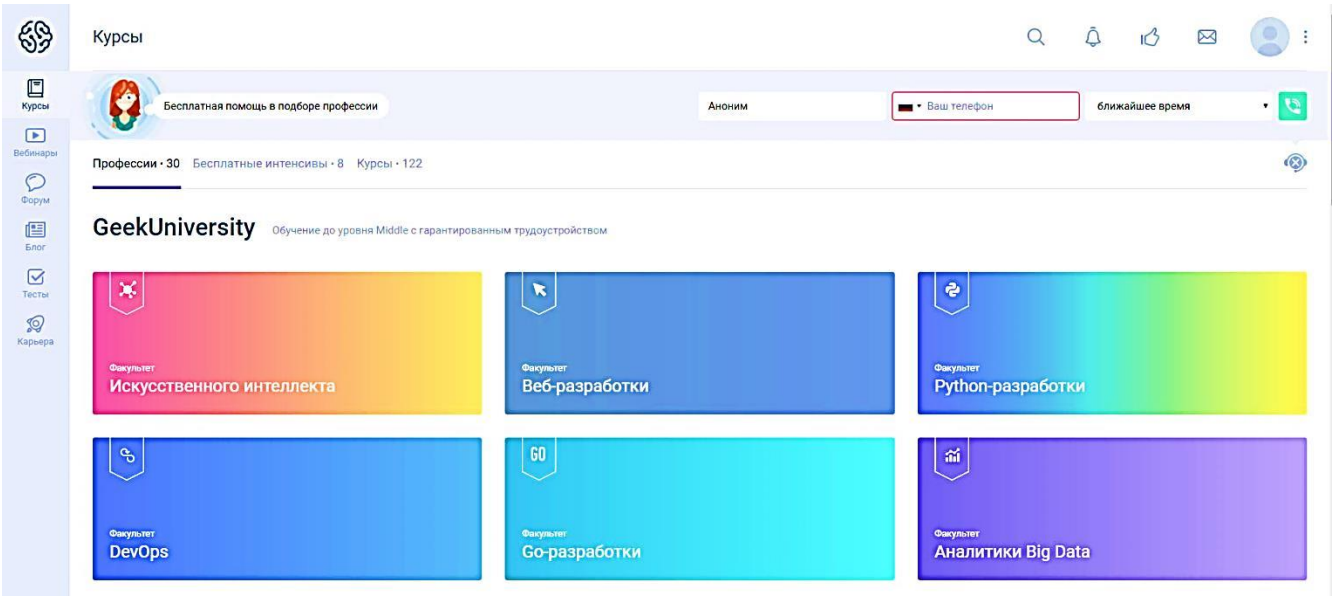


Рисунок 1.2 – Сторінка навчальних курсів системи GeekBrains

Дана сторінка містить перелік всіх можливих курсів які викладаються в цьому навчальному закладі. При натисненні на певний курс студент переходить на сторінку з описом цього курсу, відображенням дорожньої карти та плану навчання, також є можливість оплатити навчання. На мою думку на цій сторінці треба додати коротку інформацію при наведенні на певний курс, щоб студент при

ознайомленні не витрачав час на перегляд довідкової інформації про курс. Зважаючи на те, що саме після реєстрації студент потрапляє на цю сторінку.

Щодо оптимізації даної системи та швидкодії можна зробити висновок на основі рис. 1.3

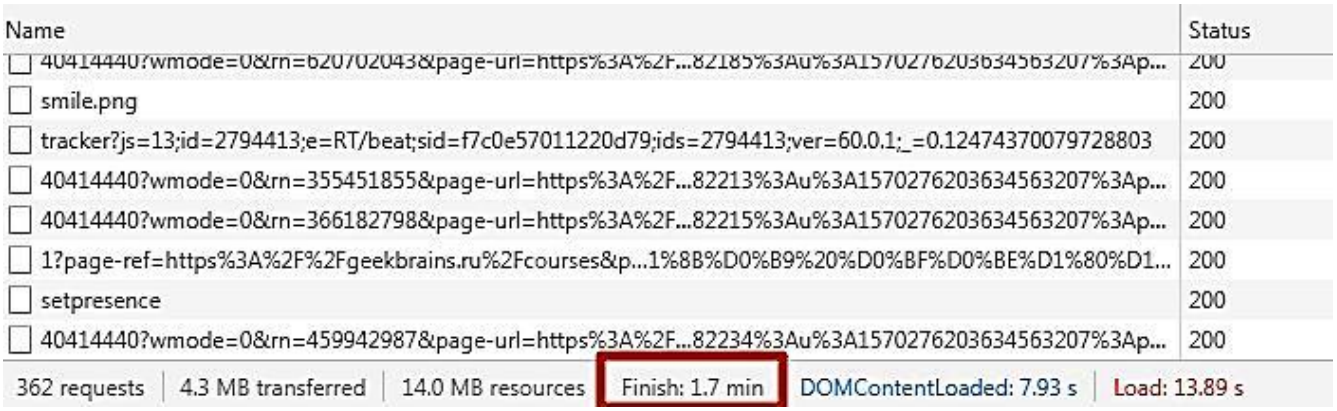


Рисунок 1.3 – Час завантаження сторінки навчальних курсів

Проаналізувавши отриману інформацію, а саме час повного завантаження сторінки майже досягає двох хвилин, що є не припустимим. Студент, котрий щойно зареєструвався та перенаправляється на цю сторінку має чекати майже дві хвилини, щоб переглянути наявні курси. На настільки довгий час завантаження сторінки впливає наявність великої кількості додаткових модулів, серед яких – модуль онлайн чату з адміністратором, модуль замовлення зворотного дзвінка та блок нагадування найближчого заходу або заняття. На мою думку від модуля зворотного дзвінка можливо відмовитися, беручи до уваги те, що є модуль онлайн чату з адміністратором. Тобто функціональне та інформаційне навантаження цих схожих модулів можна поєднати та отримати виграш у часі близько однієї хвилини.

Щодо загальної оптимізації кабінету можна зробити висновок, що сайт оптимізований досить непогано, W3C валідатор помилок не знайшов, а про оптимізацію зображень можна проаналізувати рис. 1.4

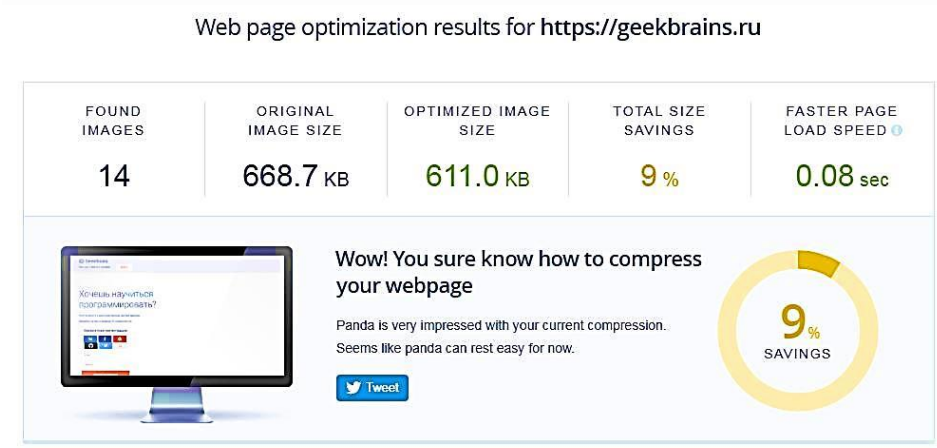


Рисунок 1.4 – Результат оптимізації картинок кабінету

Проаналізувавши результати можна зробити висновок що оптимізація зображень зроблена на достатньому рівні, а саме на 91%. Подібний результат зберігається також і на інших сторінках кабінету. Дана оптимізація є важливою тому, що сучасні сайти перенасичені зображеннями, а це на пряму впливає на швидкість завантаження сторінок, що є ключовим моментом на мобільних пристроях.

Сторінка обраних курсів, тобто тих на які записався студент виконана по принципу сторінки зі всіма можливими курсами та зображена на рис. 1.5

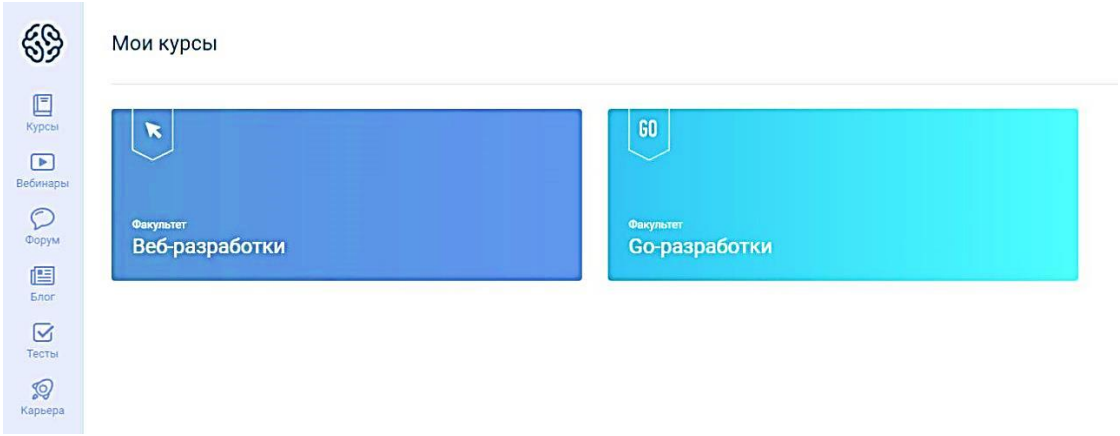


Рисунок 1.5 – Сторінка обраних курсів

Схожість даної сторінки на сторінку переліку всіх курсів означає те, що вона перейняла усі як позитивні сторони, такі як простота та чистий, не перенасичений

дизайн так і недоліки, а саме недостатність короткої інформації про курс задля скорочення часу на перегляд довідкової інформації про курс.

Сторінка з персональною інформацією виконана у вже притаманному для даної системи чистому дизайну без зайвих елементів інтерфейсу, що можна побачити на рис 1.6.

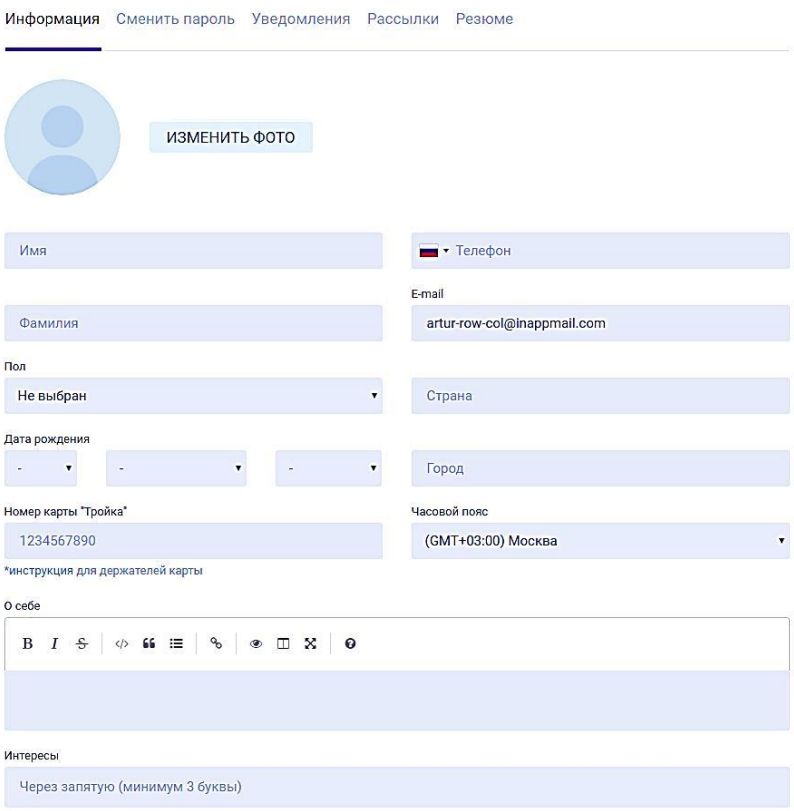


Рисунок 1.6 – Сторінка персональної інформації студента

Дана сторінка містить блоки для надання або актуалізації персональної інформації, зміни паролю, перегляду нагадувань та системних повідомлень, управління роботою розсилок повідомлень та розміщення резюме.

Провівши аналіз даної системи можна зробити висновок, що дана система має чистий, сучасний на приємний дизайн, зрозумілий та не перенасичений інтерфейс, що повинно приємно вразити студента при першому знайомстві із системою. Персональний кабінет має багато функціональних можливостей, серед яких форум, блог, тестування, вебінари на пошук роботи після завершення

навчання. До недоліків модна віднести великий час завантаження сторінок, через додаткові модулі, деякі з яких модна поєднати для оптимізації роботи.

В свою чергу не менш відома система для навчання студентів є HTML Academy, що зображена на рис. 1.7.

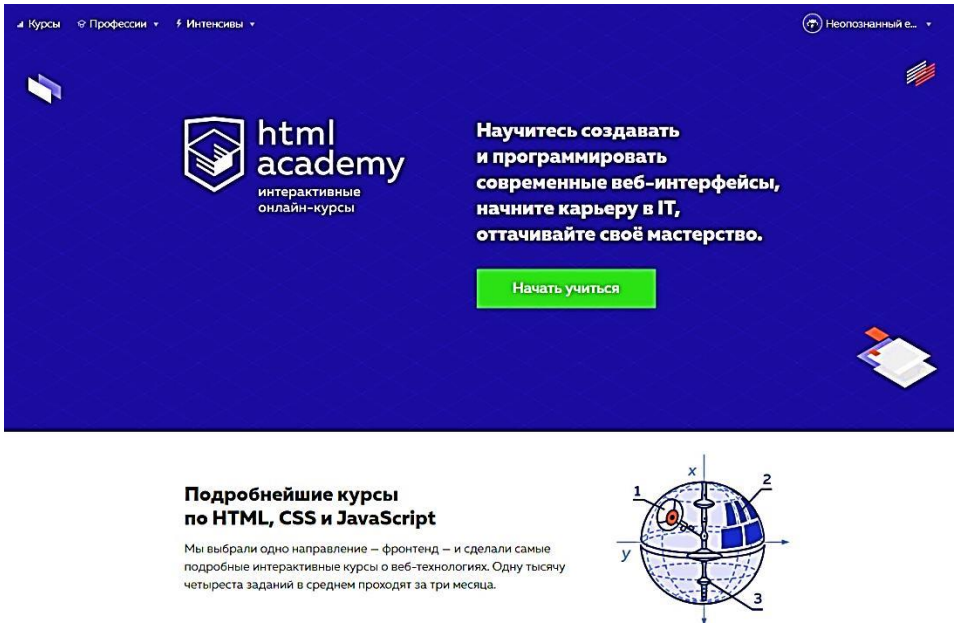


Рисунок 1.7 – Головна сторінка системи HTML Academy

Головна сторінка системи має сучасний та чистий дизайн, не перенасичений зайвими елементами. Ця сторінка досить проста, так-як містить лише логотип, назву системи та блок із актуальними новинами.

Після реєстрації або авторизації студент потрапляє на головну сторінку персонального кабінету (рис. 1.8). Дана сторінка містить верхнє горизонтальне меню із пунктами навчальних курсів, пошуку роботи та додаткових тематичних інтенсивів з різноманітних областей. Далі студенту відображений список із усіх можливих курсів, які викладаються в цьому навчальному закладі. Кожен курс має персональний логотип, назву, короткий опис, кількість навчальних блоків та завдань. При натисненні на один із цих курсів студент потрапляє на сторінку відповідного курсу, де має можливість більш детально ознайомитися з програмою навчання, викладачами та здійснити замовлення цього курсу. Дана сторінка

виконана дуже просто, без зайвих та недоречних блоків, відображена та інформація яку очікує побачити саме користувач.

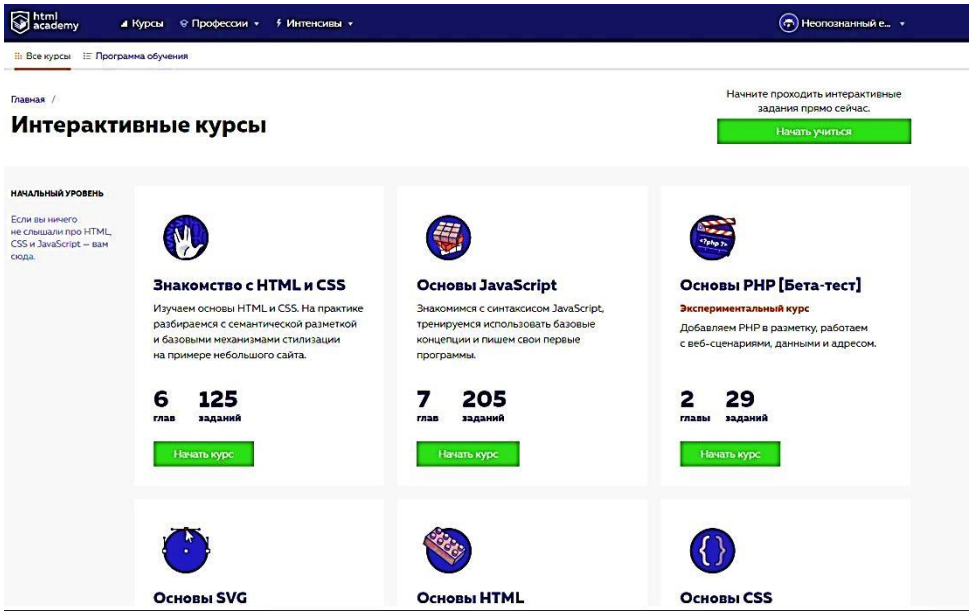


Рисунок 1.8 – Сторінка навчальних курсів

Щодо аналізу оптимізації системи можна відмітити досить високу швидкість завантаження основних сторінок (рис. 1.9). Це свідчить про те, що як серверна так і клієнтська частини дуже добре оптимізовані, відсутні модулі онлайн листування з адміністрацією. В свою чергу ці модулі винесені на окрему сторінку, що є доцільним. Висока швидкість завантаження завдячує досить обмеженому функціональному навантаженню системи, на відміну від першої розглянутої системи.

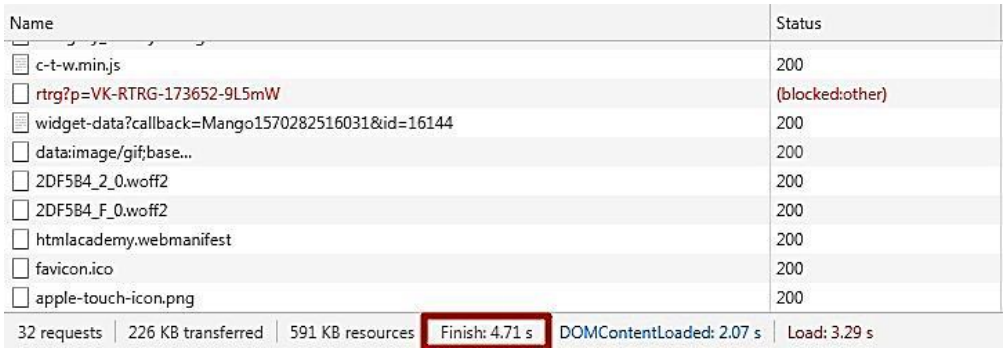


Рисунок 1.9 – Швидкість завантаження сторінки навчальних курсів

Сторінка кабінету, для зміни персональних даних теж виконана в дуже простому вигляді. Відсутня можливість завантаження резюме та перегляду

пропозицій по роботі саме для свого обраного напрямку. Проаналізувавши дану систему можна зробити висновок, що функціональні можливості досить обмежені. Система має приємний та простий вигляд. Обмежений функціонал призвів до високої швидкості завантаження сторінок.

1.3 Постановка задачі

Метою магістерської дисертації є оптимізація, підвищення швидкої та продуктивності персонального кабінету студентів для навчання робототехніці.

Для досягнення цієї мети необхідно вирішити наступні задачі:

- проаналізувати існуючі аналоги, виявивши їх переваги і недоліки та у випадку наявності останніх визначити можливі причини їх наявності;
- на основі аналізу аналогічних систем визначити подальшу концепцію систему та необхідні модулі системи;
- розробити структуру бази даних, обґрунтувати її основні функціональні блоки та виконати базове налаштування для найчастіше виконуваних запитів;
- створити моделі для зручної роботи з базою даних;
- розробити серверну частину системи з її основними модулями;
- розробити тести для автоматичного тестування як серверної так і клієнтської частин системи;
- інтегрувати систему кешування Redis, що забезпечить високий рівень швидкодії;
- розробити механізми виконання складних задач за допомогою додаткових утиліт;
- врахувати подальшу можливість масштабування системи.

Висновки до розділу

В даному розділі розглянуто область дослідження, а також найпопулярніші аналогічні системи для навчання студентів. Першою із таких систем розглянута система навчального закладу GeekBrains, яка мала досить низьку швидкість завантаження сторінок при достатній оптимізації як самої системи так і безпосередньо самого контенту. Низька швидкість завантаження обумовлена наявністю додаткових сторонніх модулів для онлайн листування з адміністрацією закладу. До переваг даної системи можна віднести велику кількість функціональних можливостей, зрозумілий користувацький інтерфейс та сучасний дизайн. Інша аналогічна система має високу швидкість завантаження сторінок персонального кабінету, це обумовлено значно скромнішими функціональними можливостями ніж у першої системи. До переваг даної системи можна віднести сучасний, чистий дизайн та високу швидкодію при досить гарній оптимізації системи.

Враховуючи всі проаналізовані переваги та недоліки обох систем визначено мету, предмет та об’єкт дослідження. Також визначено основні задачі, які необхідно виконати для досягнення зазначеної мети.

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПЕРСОНАЛЬНОГО КАБІНЕТУ

Зважаючи на те, що поставлена мета розробити та провести оптимізацію системи персонального кабінету студента курсів навчання робототехніці, то буде доцільним використання наступного переліку відповідних веб-технологій. Для розробки клієнтської частини кабінету, а саме інтерфейсу для взаємодії користувачів із системою доцільним є застосування мови розмітки документу HTML, каскадних таблиць стилів CSS з використанням препроцесору LESS та фреймворку Bootstrap. Для надання сторінкам персонального кабінету динаміки необхідно застосувати мову програмування JavaScript, а саме фреймворк React, який максимально підходить для реалізації поставленої мети. Для розробки серверної частини доцільно застосувати мову програмування PHP, а саме її фреймворк Laravel. Дана мова якомога більше підходить для реалізації поставленої мети тому, що її фреймворк має розширені можливості для роботи з базою даних, за допомогою вбудованої ORM на основі моделей даних, маршрутизацією тощо. В якості бази даних доцільно застосувати реляційну базу даних MySQL. Даний стек технологій спроможний якомога краще реалізувати всі поставлені задачі із високим рівнем швидкодії, надійності та оптимізації.

Розглянемо переваги та особливості перелічених веб-технологій.

2.1 HTML, LESS та фреймворк Bootstrap

Розгляд необхідних технологій необхідно розпочати з мови розмітки HTML.

HTML – стандартизована мова розмітки веб-документів, яка використовується в мережі Інтернет. Слід зазначити, що це не мова програмування. Текстові документи в яких розміщений код розмітки обробляється браузером, який після виконання перетворення формує відформатований документ. Структура HTML документу складається з послідовності ієрархічно розміщених елементів, які називають тегами. Тег зазвичай має початковий

елемент та кінцевий, між якими розміщується контент. В свою чергу теги можуть мати довільну кількість атрибутів, тобто властивостей відповідного тегу. Дані атрибути можуть стилізувати, розміщений в середині контент, задавати різноманітні ідентифікатори для звернення до тегу із таблиць стилів CSS або скриптів мови JavaScript. Типова структура HTML документу має наступний вигляд (рис. 2.1).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example Web Page</title>
  </head>
  <body>
    <h1>Example Page</h1>
    <p>This is a sample page.</p>
    <p>To read more, go to our <a href="info.html"</a>info</a> page.</p>
    <!--- this is a comment in the code -->
  </body>
</html>
```

Рисунок 2.1 – Базова структура HTML документу

Як можна побачити існує кореневий тег *<!DOCTYPE html>*, який вказує на тип документу та його версію, а саме HTML5. Далі по ієрархії документу розміщений тег *<html>*, який є головним контейнером, який об'єднує в собі всю розмітку документу. Наступними розміщені теги *<head>* та *<body>*, які знаходяться на одному рівні ієрархії об'єктної моделі документу. В свою чергу тег *<head>* відповідає за розміщення тегів, таких як мета теги, сервісні теги для роботи браузеру, теги додавання стилів та скриптів. Тег *<body>* виконує функцію зберігання вмісту веб-сторінки, яке відображається у вікні браузеру. В ньому розміщається велика кількість тегів, таких як *<p>* – тег параграфу, *<a>* – тег відображення посилання на будь-який ресурс в мережі, *<h1>* – тег заголовку першого рівня, *<img>* – відображення зображень та інші. В свою чергу всі ці теги можуть бути поміщені один в одного, скомпоновані різноманітними способами та використані в залежності від поставленої задачі. На даний момент використовується версія HTML5, яка розширила перелік стандартних тегів

специфічними, які окрім звичайних функцій розмітки виконують відповідну семантичну функцію, що спрощує SEO оптимізацію документу та більш зрозуміло структурує документ.

Каскадні таблиці стилів або CSS – це формальна мова опису відображення документу. За допомогою CSS можна стилізувати будь який елемент веб-документу. Наприклад можливо задати висоту, ширину або відступі в довільному блоці або колір та гарнітуру шрифту для параграфу або заголовку тощо. Для того, щоб задати стилізування певного елементу необхідно вказати селектор по якому буде обиратися елемент, в якості якого може бути використаний *id* елемента, клас, або безпосередньо сам HTML тег. Далі вказуються необхідні властивості та їх значення.

Для того щоб підвищити продуктивність написання CSS коду та забезпечити його гнучкість та масштабованість застосовують препроцесори. Вони мають схожий синтаксис на CSS але мають більш розширенні можливості такі як використання змінних, міксинів, доповнення класів, рівнів вкладення тощо. Для написання стилів в даному проекті необхідно застосувати препроцесор LESS. Для того щоб браузер міг розпізнати та застосувати стилі, то код написаний за допомогою препроцесору необхідно скомпілювати в CSS за допомогою додаткових бібліотек або розширень. Приклад коду написаного за допомогою препроцесора та еквівалентний код CSS зображено на рис. 2.2.

Як можна побачити за допомогою препроцесору набагато зручніше писати код стилів, а саме за рахунок використання синтаксису вкладень та посилань на батьківський елемент. Код написаний за допомогою використання препроцесорів набагато краще аналізувати, масштабувати та підтримувати.

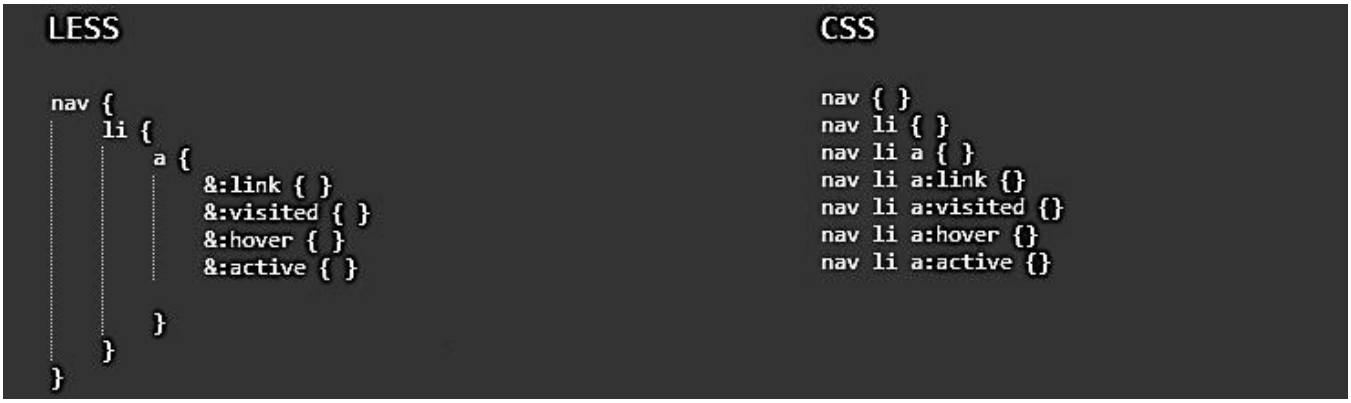


Рисунок 2.2 – Порівняння синтаксису CSS та LESS

Bootstrap – це найпопулярніший в наш час CSS фреймворк, який значно спрощує та пришвидшує розробку сучасних веб-додатків. За допомогою нього можна швидко розробити верстку проекту, та за рахунок вже розроблених компонентів, таких як кнопки, списки, елементи форм та інші спроектувати веб-додатки. Використання даного засобу зводиться до використання вже написаних класів та деяких атрибутів до власних HTML елементів.

2.2 JavaScript та фреймворк React

JavaScript – це об’єктно-орієнтована мова програмування, яка є однією із версій мови ECMAScript. Вона широко застосовується для надання веб-сторінкам динаміки, а саме такі елементи як слайдери, таби, модальні вікна та інші. Програми написані на цій мові називають скриптами. Вони можуть вбудовуватися в розмітку веб-документу та виконуватися при завантаженні самої веб-сторінки. В наш час ця мова може застосовуватися та виконуватися не тільки в браузері, а і на сервері та на будь-якому іншому пристрої який має спеціальний програмне ядро. За допомогою JavaScript можливо додавати нові HTML елементи на сторінку, змінювати або видаляти вже існуючі, реагувати на дії від користувачів, такі як натиснення клавіш на клавіатурі або маніпулювання «мишкою», завантажувати файли та відправляти запити на віддалені сервери мережі, отримувати та застосовувати *cookie* файли, запам’ятовувати введені дані на стороні користувача та багато іншого.

Для реалізації поставленої мети для даного проекту не достатньо можливостей чистої мови програмування JavaScript. Тому вирішено застосувати фреймворк React, який спеціалізується на створенні в першу чергу односторінкових додатків, що і підходить для розробки персонального кабінету студента курсів робототехніки. React набагато спрощує створення користувацьких інтерфейсів, необхідно лише описати різноманітні стани кожної частини додатку. Цей фреймворк заснований на компонентному підході, тобто весь додаток розподіляється на окремі логічні блоки або компоненти, які мають свою логіку та стан. За допомогою стану компоненту React досить просто у відповідь на його зміну зможе відобразити змінену частину компоненту не перезавантажуючи повністю весь додаток. Цей підхід заснований на такому явищі як віртуальний DOM, тобто створюється віртуальна копія об'єктної моделі документу і у випадку зміни стану або React порівнюючи оригінальний та віртуальний DOM перезавантажує лише змінену частину додатку, що позитивно впливає на оптимізацію та швидкодію. Синтаксис чистого JavaScript та фреймворку React дуже відрізняються (рис 2.3).

Перша відмінність яка кидається в очі це застосування JSX, це мова яка дозволяє використовувати HTML теги безпосередньо в самому коді та працювати з ним як довільними елементами. Інша відмінність – це використання методів життєвого циклу компоненту. Це методи за допомогою яких можна відслідкувати стан певного компоненту у будь який момент його роботи, а саме до відображення на сторінці, після відображення, в момент першого відображення, після оновлення та інші.

```
import React, { Component, PropTypes } from 'react';
import LinkButton from './LinkButton.jsx';

class ButtonList extends Component {

  componentDidMount() {
    this.props.updateStateFromServer();
  }

  render() {
    const { buttons } = this.props.buttonListState;
    const buttonNames = Object.keys(buttons);

    return (
      <div className="NavButtonContainer">
        {buttonNames.map(name => (
          <div key={name} className="NavButton"><LinkButton name={name} urls={
            buttons[name]} /></div>
        ))}
      </div>
    );
  }
}

export default ButtonList;

```

Рисунок 2.3 – Синтаксис фреймворку React

Саме компонентний підхід дозволяє слідкувати за тим які модулі підключені на сторінках, що є передумовою до оптимізації використовуваної пам’яті, так як після закінчення використання певного компоненту, він фізично вивільняється з оперативної пам’яті.

2.3 PHP та фреймворк Laravel

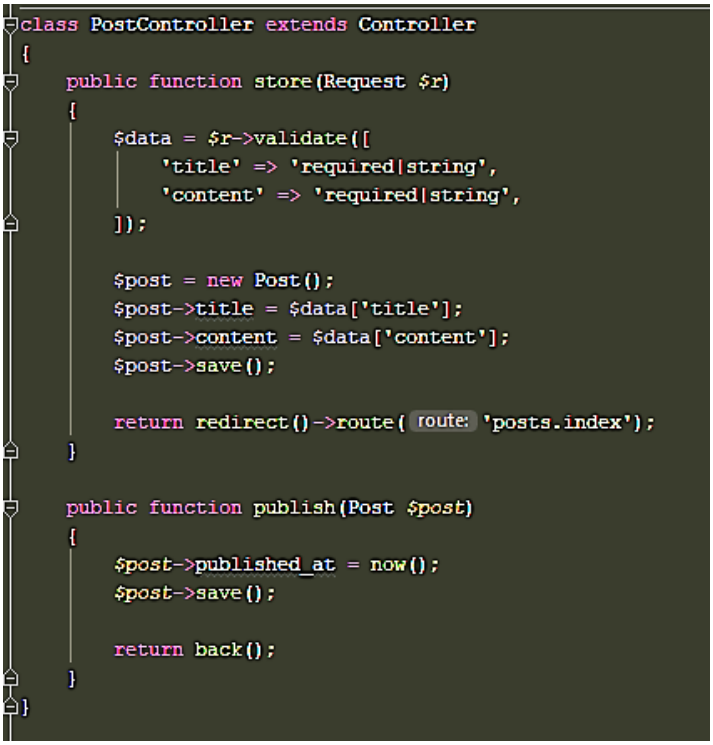
PHP – це мова програмування з відкритим вихідним кодом, яка спеціально створена для розробки веб-додатків різного рівня складності. Найголовнішою відмінністю PHP від мови JavaScript та його фреймворків є те, що PHP-скрипти виконуються безпосередньо на сервері, генеруючи HTML сторінку та передаючи її клієнту. Отримавши з серверу HTML документ клієнт не може дізнатися який PHP – скрипт його виконав тому, що він є результатом його виконання. Дана мова є мовою з динамічною типізацією, яка не потребує вказувати тип даних змінної та взагалі її об’являти. Перетворення між скалярними типами даних здійснюються у неявному вигляді. PHP має наступні типи даних: скалярні типи даних, до яких

можна віднести цілі числа, з плаваючою крапкою та строкові типи даних та логічні, нескалярні, серед яких об’єкти, масиви та зовнішні ресурси.

Свою прихильність користувачів отримав через підтримку великої кількості вбудованих можливостей, а також різноманітних додаткових модулів для розробки. До таких них можна віднести модуль для HTTP авторизації користувачів, модуль обробки та збереження файлів, робота з *cookies* та відкритими сесіями користувачів, автоматичне вилучення POST та GET заголовків із тіла відповідного запиту та підтримку великої кількості систем управління базами даних, таких як MySQL, Oracle, PostgreSQL, Redis, SQLite та багатьма іншими.

Для реалізації поставленої задачі вирішено використовувати фреймворк PHP, а саме Laravel. Це обумовлено тим, що з його використанням набагато простіше, зручніше та якісніше розробляти веб-додатки, за допомогою вже готових модулів авторизації та автентифікації користувачів, маршрутизації, шаблонізаторів веб-сторінок, модулів для роботи з електронною поштою, сучасним RESTApi тощо. Всі необхідні додаткові модулі можна завантажити із офіційного репозиторію. Приклад коду написаного з використанням фреймворку Laravel зображено на рис. 2.3.

Додаткові модулі встановлюються та підключаються за допомогою менеджеру пакетів *Composer*. За допомогою вбудованої ORM Eloquent представлена можливість створювати моделі для зручного створення та роботи з базою даних та маніпулювання даними всередині БД. Для запобігання завантаження не використовуваних компонентів внаслідок їх ручного підключення реалізований механізм автоматичного завантаження класів. Механізми зручного оновлення та розгортання веб-додатків а також бази даних досягається за рахунок використання міграцій, схожу за принципом роботи на систему контролю версій.



```
class PostController extends Controller
{
    public function store(Request $r)
    {
        $data = $r->validate([
            'title' => 'required|string',
            'content' => 'required|string',
        ]);

        $post = new Post();
        $post->title = $data['title'];
        $post->content = $data['content'];
        $post->save();

        return redirect()->route( route: 'posts.index');
    }

    public function publish(Post $post)
    {
        $post->published_at = now();
        $post->save();

        return back();
    }
}
```

Рисунок 2.4 – Синтаксис фреймворку Laravel

Для розробки веб-сторінок зручно використовувати вбудований шаблонізатор Blade, за допомогою якого можна розбити веб-сторінку на окремі модулі та використовувати деякі блоки повторно за допомогою спеціального синтаксису. В свою чергу присутня можливість використовувати вбудовані консольні команди або створені власноруч для оновлення, розгортання тощо за допомогою інтерфейсу Artisan.

До переваг даного фреймворку можна віднести гнучку систему маршрутизації, простий синтаксис API фреймворку, зручні механізми обробки помилок та виключень, вбудований механізм авторизації та автентифікації користувачів. Також без встановлення додаткових модулів Laravel може запропонувати механізми кешування веб-додатків за допомогою технологій Redis та Memcached та методи для файлового кешування. Laravel має простий API інтерфейс для популярних бібліотек, наприклад таких як SwiftMailer з підтримкою драйверів SMTP, sendmail та Amazon SES. Також Laravel має зручний інтерфейс до сервісів білінгу. Можна відмітити надійний захист бази даних від вразливостей,

таких як XSS та CSRF. Також Laravel в базовій конфігурації має деякі готові модулі для тестування веб-додатку.

Щодо малочисельних недоліків, то можна відмітити відсутність зворотної сумісності між версіями фреймворку, недосконаліть додаткового синтаксису та нелогічне розміщення деяких файлів та папок, а саме модель *User.php*, що розміщена в */app*, замість логічного розміщення в */app/Models* та каталог */resourses*, який розміщений в кореневій директорії а не в */app/resourses*. Також варто зазначити довший час розробки простих сайтів за допомогою фреймворку Laravel ніж на більш простих фреймворках або на чистому PHP.

2.4 Система управління базами даних MySQL

MySQL – це система управління реляційними базами даних, яка заснована на моделі клієнт-сервер та має відкритий вихідний код. Дана СУБД використовує для виконання запитів мову SQL. Слід зазначити, що дана система управління є крос-платформною та працює на всіх операційних системах, таких як Windows та Linux. Ядром даної системи управління для обробки вхідних запитів та їх виконання, а також фільтрації, агрегації, сортування та оновлення даних – є сервер MySQL. Вибір саме цієї СУБД обумовлений її високою швидкодією безпосередньо для веб-додатків. Панель адміністратора даної СУБД зображена на рис. 2.4.

За допомогою даної системи управління базами даних можливо створити таблиці типу InnoDB та MarinaDB. Головною відмінністю цих типів є те, що таблиці типу InnoDB дозволяють створювати та використовувати зовнішні ключі. Щодо таблиць типу MarinaDB, то така можливість відсутня. Зовнішні ключі необхідні в таблицях для забезпечення умов цілісності даних. Під час кожного нового запису до бази даних відбуваються перевірка на відсутність перешкоджань у зв’язках та зовнішніх ключах таблиць. Щодо таблиць типу MarinaDB, то за рахунок відсутності зовнішніх ключів, розширюється область їх використання, а

саме рекомендоване використання у високонавантажених та складних веб-додатках та системах, основним вектором розробки яких є високий рівень швидкодії.

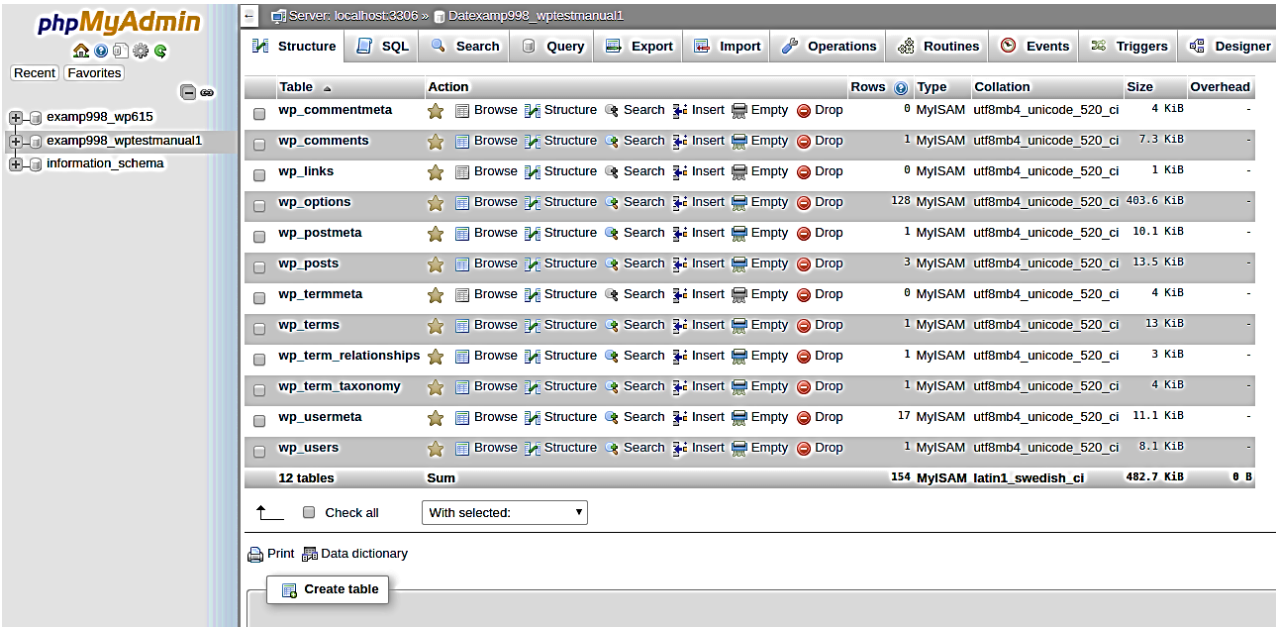


Рисунок 2.4 – Панель адміністратора СУБД MySQL

Високий рівень швидкодії при використанні таблиць такого типу досягається саме через відсутність зовнішніх ключів, а саме через відсутність перевірок при додаванні нових даних до БД. До недоліків такого типу таблиць можна віднести можливе дублювання або зберігання некоректних даних, що призводить до застосування у досить вузькому колі задач. У багатьох типових веб-додатках використання таблиць типу MarinaDB – є не доцільним. Використання в якості системи управління базами даних саме MySQL обумовлено можливістю додаткового копіювання та розподілу даних по таблицям, що дозволяє збільшити швидкодію веб-додатків при обробці великої кількості даних у високонавантажених системах. Може виникнути ситуація коли одна таблиця буде містити дуже багато записів, що знизить швидкість пошуку в ній. Тому саме метод розподілення даних із однієї великої таблиці на декілька інших пришвидшує пошук по записам. Важливою перевагою MySQL є те, що для роботи з нею необхідно вивчати додатковий синтаксис. MySQL дозволяє виконувати запити на

додавання, оновлення, вибірку, групування, сортування, об’єднання та агрегацію даних можливостями та командами мови SQL.

Висновки до розділу

В даному розділі розглянуто та обрано необхідні технології та програмні засоби для досягнення поставленої мети. Тому обрано мову розмітки веб-документів HTML для створення каркасу користувацького інтерфейсу веб-додатку. Також обрано фреймворк каскадних таблиць стилів CSS – Bootstrap 4 для оформлення веб-додатку відповідно до розробленого дизайну та макету. Написання стилів вирішено розробляти за допомогою препроцесору LESS, що досить пришвидшить та підвищить продуктивність роботи за допомогою використання змінних, міксинів та розширень. Щодо написання логіки користувацького інтерфейсу вирішено застосувати мову програмування JavaScript та безпосередньо його фреймворк React. Застосування останнього обумовлено високим рівнем швидкодії та надійності. Високий рівень швидкодії системи досягається за рахунок використання модульної системи компонентів, коли весь веб-додаток розподіляється на окремі компоненти, які зручно розробляти, редагувати та тестувати окремо від усього додатку. Щодо мови програмування для розробки серверної частини додатку обрано PHP та її фреймворк Laravel. Використання останнього обумовлено високим рівнем захищеності додатку та БД на його основі від більшості веб-загроз, велику кількість додаткових модулів та направленість використання на високонавантажені веб-додатки. В якості СУБД для реляційної бази даних вирішено використати MySQL, за рахунок її направленості на веб-системи, високу швидкодію та функціональність.

РОЗДІЛ 3. РОЗРОБКА ПЕРСОНАЛЬНОГО КАБІНЕТУ СТУДЕНТА

3.1 Архітектура веб-системи

Персональний кабінет студента курсів навчання робототехніці складається з десяти основних функціональних модулів, які забезпечують її повноцінне функціонування (рис. 3.1).

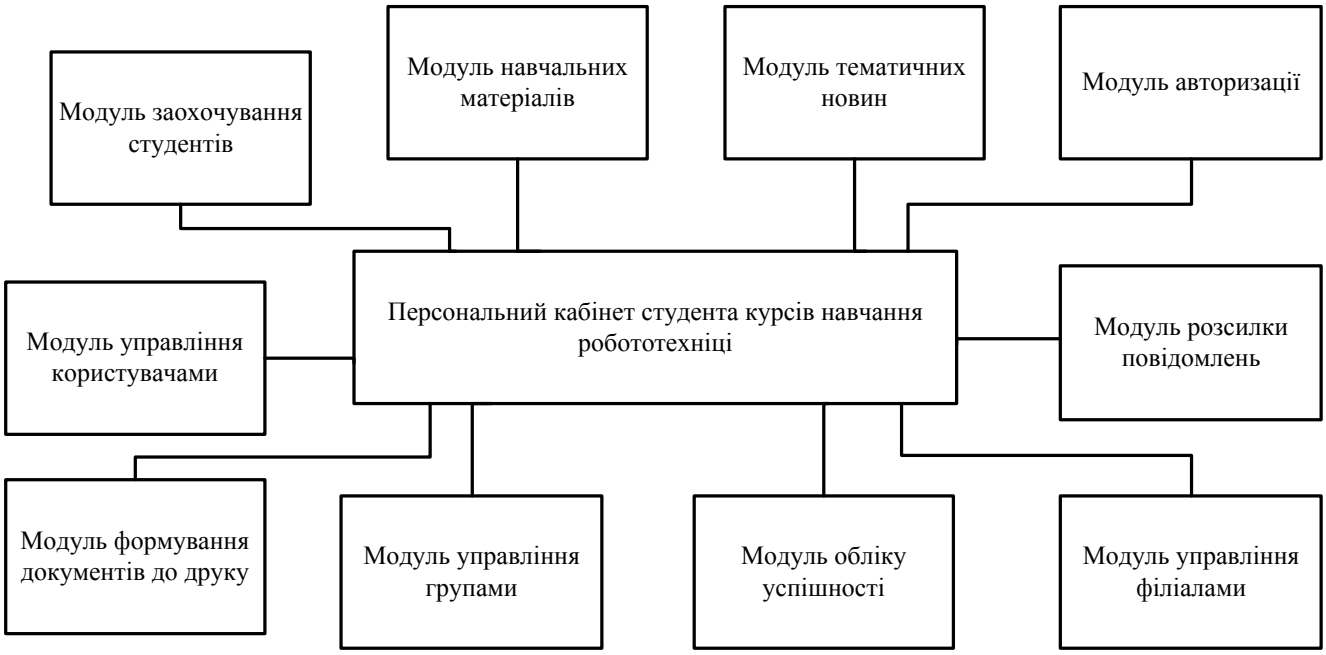


Рисунок 3.1 – Структура персонального кабінету студента

Розгляд даної структури доцільно почати з модуля управління авторизацією системи. Даний модуль відповідає за відображення форми для входу у систему персонального кабінету, тобто те, що користувач бачить при бажанні відвідати кабінет. Він проводить авторизацію користувачів на основі їх унікальних даних, таких як логін або електронна пошта та пароль, а також дає можливість запам’ятати пароль для більш швидкого та зручного подальшого проходження авторизації, провести процедуру відновлення забутого паролю, видачу унікального ідентифікатора користувача при успішному проходженні авторизації, реєстрацію нових користувачів. За допомогою функції запам’ятовування паролю

браузер зберігає в кеші не тільки пароль а і унікальний ідентифікатор користувача, отриманий при створенні нової сесії. Процес відновлення забутого паролю здійснюється за такою процедурою, а саме користувач вводить свою адресу електронної пошти і у випадку якщо така пошта справді зареєстрована у системі, він отримує на цю адресу лист із активним посиланням, перейшовши по якому він потрапляє на форму введення нового паролю. Беручи до уваги той факт, що паролі зберігаються в базі даних у зашифрованому вигляді, тобто отриманий під час реєстрації пароль, модуль авторизації зашифровує його з використанням алгоритмів шифрування та зберігає в БД. Тому відновити забутий пароль просто неможливо, бо процедура шифрування паролю є односторонньою. Функція реєстрації нових користувачів полягає у заповненні форми з полями: ім’я, прізвище, електронна пошта, номер телефону, пароль. Слід зазначити, що поле номеру телефону та електронна пошта є унікальними для системи в цілому, тобто не можуть повторюватися. Процес видачі унікального ідентифікатора полягає у тому, що при успішному проходженні авторизації створюється нова сесія для користувача і в результаті цього він отримує ключ який зберігається в кеші браузера і необхідний для подальшого знаходження в системі та проведення будь-яких операцій.

Успішно завершивши процес авторизації користувач потрапляє головну сторінку персонального кабінету, на якій виводиться блоки з модуля тематичних новин. Даний модуль призначений для відображення студентам тематичних новин в залежності від їх уподобань або відвідуваних ними курсів. Оновленням, додаванням або видаленням новин займається адміністратор або модератор персонального кабінету. На цій сторінці зазвичай відображається чотири блоки з новинами, структура яких має заголовок, зображення попереднього перегляду, короткий опис новини та кнопку для перегляду повної статті.

Кожен зареєстрований користувач має можливість змінити свої персональні дані, які він вказав на етапі реєстрації, змінити адресу електронної пошти та пароль у розділі головного меню – *Мій профіль*. В свою чергу після проходження навчального курсу студенту з'являється можливість додати своє резюме для можливого влаштування на стажування у певну організацію на основі отриманих знань.

Підсистема модулів адміністрування персонального кабінету складається з модулю розсилки повідомлень, управління користувачами, формування документів до друку, управління групами та управління філіалами. Одним із основних модулів цієї підсистеми – є модуль управління користувачами. До основних функцій даного модуля можна віднести додавання, редагування, видалення та перегляд інформації про користувачів системи. Щодо ролі користувач, то в системі передбачено роль *Адміністратор*, *Студент*, *Абітурієнт*, *Викладач*. Щодо ролі *Адміністратор* – користувач, в компетенцію якого входять всі можливі дії над іншими користувачами, групами, навчальними матеріалами, філіалами тощо. Тобто він може додавати, оновлювати, видаляти інформацію про студентів, навчальні групи, курси, філіали, проводити так ж операції над навчальними матеріалами тощо. *Студент* – користувач, який обрав хоча б один курс, та приступив до навчання. В свою чергу він має доступ до навчальних матеріалів курсу, перегляду своєї успішності, бонусів та персональної інформації. *Абітурієнт* це користувач, який зареєструвався у системі але ще не визначився із напрямком навчання та має доступ тільки до переліку можливих курсів для навчання. *Викладач* – це користувач який закріплюється за однією або декількома групами студентів та має доступ до розміщення та редагування навчальних матеріалів своїх курсів, перегляду інформації про студентів, оцінюванню студентів, розміщенню інформаційних статей з тематики своєї спеціалізації, редагування персональної інформації та зміни адреси електронної пошти та

паролю. Щодо модулю управління групами, то доступ до нього має тільки адміністратор. За допомогою нього адміністратор може додавати, оновлювати, видаляти навчальні групи, додавати студентів до них, закріплювати відповідного викладача. Модуль управління філіалами призначений для додання, оновлення інформації про філіали навчального закладу. Сутність філіал необхідна для спрощеного сортування навчальних груп, які закріплені за філіалами та більш зручного розмежування ресурсів між ними. Щодо модулю розсилки повідомлень, то основною метою його є налагодження процесу листування між адміністрацією та студентами, а також адміністрацією та викладачами навчальних курсів. В свою чергу існує можливість листування між викладачами та студентами відповідних навчальних груп. Викладачі можуть повідомляти студентам різноманітні організаційні моменти або передавати додаткові навчальні матеріали. А адміністратор може зручно листуватися із викладачами, робити розсилки певним групам студентів або всім користувачам одразу. І останнім модулем даної підсистеми – є модуль формування документів для друку. Даний модуль призначений для формування електронних документів для перегляду або друку. Документи можуть стосуватися успішності окремих груп або студентів, інформації про будь-якого користувача або навчальну групу.

Модуль обліку успішності студентів призначений для проведення обліку успішності студентів навчальних груп за певне заняття. Структура даного модулю складається з сукупності блоків занять, які об’єднані на основі певної групи та навчального курсу. Викладач на початку кожного заняття створює новий урок. Йому відображається список студентів даної групи, назва, навчальний курс та дата проведення заняття. Протягом заняття викладач виставляє оцінки відповідно до рівня успішності студента. Рейтингова система оцінювання п’яти бальна. За відвідування кожного заняття студент отримує 1 бал, а за відсутність 0 балів. За виконане домашнє завдання він отримує 2 бали. Останні 3 бали студент може

заробити за активну участь в процесі проходження заняття. В свою чергу адміністратор на сторінці обліку успішності застосувавши відповідні фільтри може переглянути створені сутності занять для певних груп відповідних курсів. Відкривши будь-яку сутність заняття, адміністратор може відредагувати всі можливі параметри, при допущенні помилок з боку викладача. Також на цій сторінці викладач та адміністратор мають можливість переглядати успішність певних груп за певний період або за весь курс в цілому. Присутня можливість гнучкого функціоналу фільтрування та виводу документу на друк.

Щодо модулю навчальних матеріалів, то він призначений для надання доступу адміністратору та викладачу для завантаження та редагування документів. В свою чергу студент має доступ на перегляд та вивантаження для матеріалів тих курсів, які він відвідує. Модуль представлений у вигляді зручного інтерфейсу у вигляді дерева каталогів, і корні якого розміщені папки всіх навчальних курсів. Доступ відбувається безпосередньо з браузеру, тобто відсутня необхідність мати доступ до файлової системи веб-сервера додатку. У директорії відповідного курсу розміщені папки з усіма заняттями та додатковими завданнями. В папках відповідних занять розміщені файли навчальних матеріалів. Дозволені типи файлів для навчальних документів наступні: *.png, .jpg, .doc, .docx, .pdf, .djvu, .pptx* та *.ppt*. Такі обмеження обумовлені налаштуванням безпеки усього веб-додатку. Директорія кожного заняття із усіма необхідними документами з’являється для доступу студента безпосередньо після закінчення відповідного заняття.

Модуль заохочення студентів являє собою сторінку, яка містить блоки із заохочувальними подарунками. Кожен студент на заняттях отримує певні бали, які додаються протягом всього проходження навчального курсу. Ці бали відображені у верхній частини персонального кабінету студента. Студент, знаючи свою кількість балів може відвідати сторінку заохочень та обрати ту річ яку він

хоче. Кожен товар коштує певну кількість балів, засновану на реальній вартості цього предмету. Після звернення студента до адміністратора з проханням про отримання заохочення, він перевіряє кількість заохочувальних балів студента, видає йому обраний предмет та списує визначену кількість балів. Адміністратор має можливість додавати, редагувати та видаляти як вартість товару так і інформацію про нього.

3.2 Структура бази даних

База даних персонального кабінету студента навчання робототехніці містить 12 таблиць для збереження інформації від відповідних модулів системи. Кожна таблиця у своїй назві містить префікс *lc_*, наявність якого значно спрощує інтеграцію персонального кабінету із іншими системами. При розгортанні веб-додатку на робочому сервері, де можуть працювати інші системи, вивантажується база даних і за рахунок цих унікальних індексів виключається можливість конфлікту вже існуючих таблиць із таблицями персонального кабінету. Все що залишається після завантаження бази даних – це виконати міграції та сіди за допомогою консолі та спеціальних команд фреймворку Laravel. Міграції – це клас із описом тих полів та зав’язків у вигляді зовнішніх ключів які необхідно створити. Міграції значно спрощують процес підтримки та розгортання веб-додатків на веб-серверах (рис. 3.2).

Кількість міграцій повинна відповідати спроектованій кількості таблиць бази даних. За допомогою вбудованих можливостей фреймворку можна запускати вибірково міграції, декілька або всі по черзі. При виконанні всіх міграцій необхідно врахувати наявність зовнішніх ключів у таблицях та виконувати спочатку міграції таблиць без зовнішніх ключів. Далі виконання відбувається чітко по архітектурі БД, щоб не виникало такого випадку, що застосовується міграція таблиці із зовнішнім ключем на ту таблицю яка ще не створена.

```
class CreateLessonTable extends Migration
{
    public function up()
    {
        Schema::create('lc_lessons', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->date('date');
            $table->unsignedInteger('group_id');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('lc_lessons');
    }
}
```

Рисунок 3.2 – Приклад створення міграцій для таблиці БД

В такому випадку фреймворк виведе помилку у консоль та зупинить виконання тих міграцій, що залишилися. Такі ж правила необхідно враховувати при розробці міграцій для таблиць.

В свою чергу застосування механізму *seeds* необхідно для заповнення розробленої бази даних тестовими даними, для більш зручного проектування веб-додатку в цілому. На кінцевому етапі розробки системи за допомогою цього механізму виконується очистка бази від тестових даних і вона готова для заповнення реальною інформацією (рис. 3.3).

Як можна побачити цей механізм являє собою створення сукупності класів, які повертають асоціативний масив, що містить структуру ключ-значення. Процес заповнення таблиць тестовими даними запускається в ручному або автоматичному режимі після закінчення процесу виконання міграцій, що вказує на певну послідовність виконання механізму *seeds*.

```
DB::table(CreateLcCoursesTable::TABLE_NAME)->insert([
    'name' => 'Web design',
    'weight' => 1,
    'parent_id' => null
]);

DB::table(CreateLcCoursesTable::TABLE_NAME)->insert([
    'name' => 'Frontend',
    'weight' => 4,
    'parent_id' => null
]);

DB::table(CreateLcCoursesTable::TABLE_NAME)->insert([
    'name' => 'Development',
    'weight' => 7,
    'parent_id' => null
]);
```

Рисунок 3.3 – Приклад створення механізму *seeds*

3.2.1 Структура таблиць та їх зовнішні зв’язки

Архітектура бази даних персонального кабінету студента зображена на рис. 3.4. Розгляд таблиць бази даних персонального кабінету доцільно розпочати з таблиці *lc_users*, яка необхідна для зберігання основної персональної інформації про користувача системи. Вона містить в собі наступні поля: ім’я, прізвище, адреса електронної пошти, номер телефону, статус користувача, фотокарточка, його оцінки та додаткові сервісні поля, які містять інформацію про створення користувача тощо.

Таблиця *lc_users_data* призначена для зберігання додаткової інформації про користувача, а саме додаткового номеру телефону, адреса, особливі відмітки тощо. Дана таблиця розроблена таким чином, щоб не залежати від кількості полів за допомогою асоціативного масиву, тобто ключ – значення. Кожен користувач може мати довільну кількість додаткових параметрів. За допомогою розділення основної таблиці користувачів та таблиці з додатковими параметрами вдалося досягти перенасичення та розростання основної таблиці в подальшому.

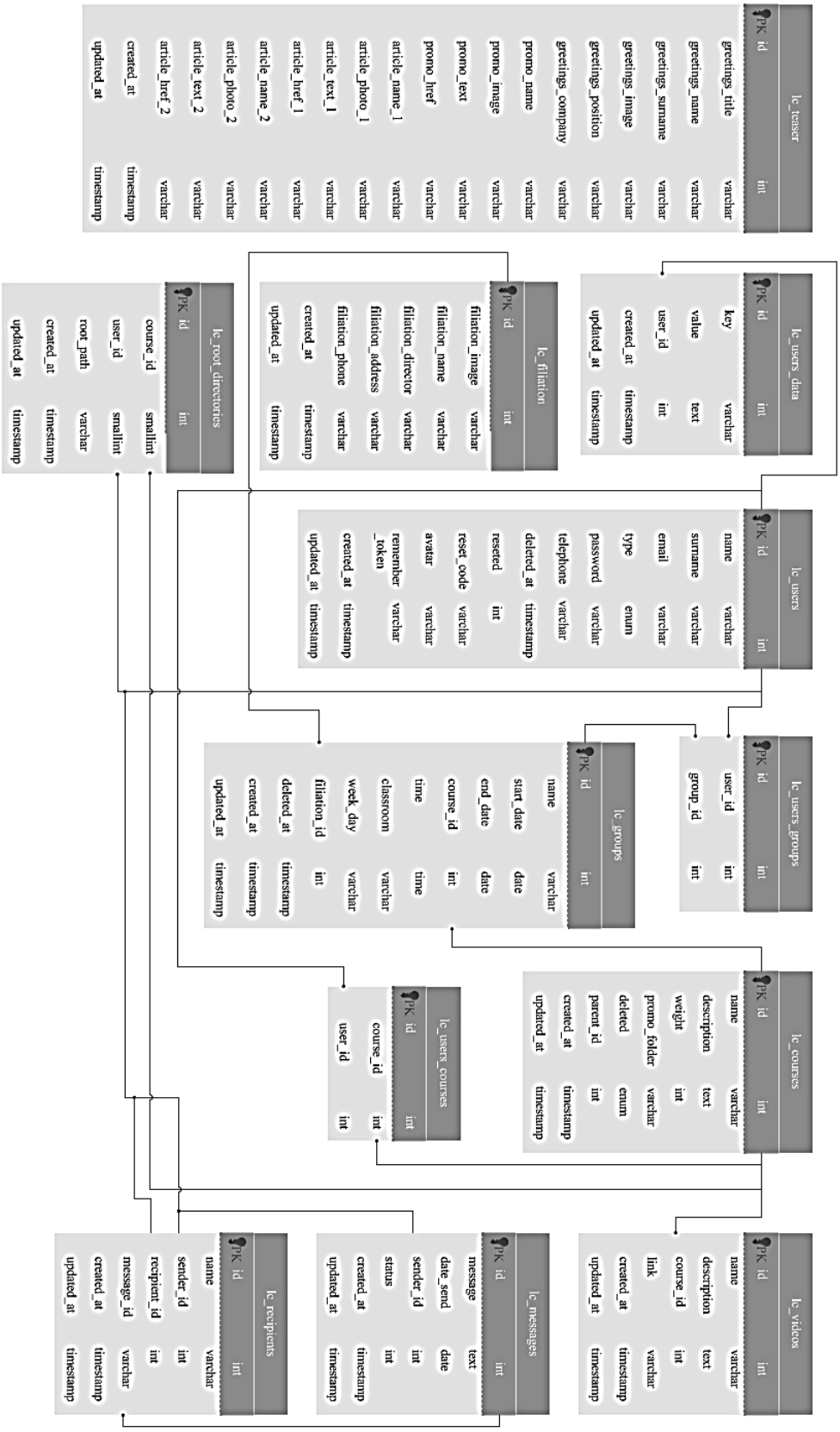


Рисунок 3.4 – Структура бази даних персонального кабінету

Таблиця навчальних курсів *lc_courses* призначена для зберігання даних про всі наявні курси у відповідному навчальному закладі. Дана таблиця містить такі поля, як назва курсу, його опис, актуальність, зображення та ідентифікатор на курс який знаходиться вище по ієрархії, для створення деревовидної архітектури курсів. Параметр актуальність курсу необхідний для зберігання значення від 0 до 1, внаслідок цього чим більший даний параметр тим вище вірогідність запропонування цього курсу новому студенту на головній сторінці персонального кабінету.

Проміжна таблиця *lc_users_courses* призначена для зберігання ідентифікатора користувача та ідентифікатора відповідного курсу. Необхідність такої таблиці викликана реалізацією зв'язку багато до багатьох. Це пояснюється тим що один студент може відвідувати безліч курсів, а в свою чергу на один курс може бути записано багато студентів.

Таблиця *lc_groups* необхідна для збереження інформації про навчальні групи, яка містить наступні поля: назва групи, дата початку та кінця навчального курсу, філіал освітнього закладу, аудиторія, час занять, список студентів, а також напрям та курс. Оскільки в даній таблиці за реалізацією також використовується зв'язок багато до багатьох, тому по аналогії до таблиці користувачів *lc_users_data*, розроблена таблиця *lc_users_groups*, яка містить ідентифікатори групи та користувача. Використання такого зв'язку пояснюється тим, що один студент може навчатися у багатьох групах, а також в одній групі може навчатися багато студентів.

Проста таблиця *lc_teasers* призначена для збереження інформації про блоки новин з головної сторінки кабінету. Вона містить наступні поля: назва, короткий опис, зображення, посилання на повну статтю, а також сервісні параметри: дата створення блоку новин та дата останнього оновлення його інформації.

Таблиця філіалів навчального закладу *lc_filiation* необхідна для збереження інформації про наявні філіали освітнього закладу. Вона містить наступні поля: назва філіалу, зображення, ФІО керівника, адреса та номер телефону, також сервісні параметри: дата створення та останнього оновлення інформації про філіал.

Таблиця товарів для заохочення найуспішніших студентів *lc_presents* призначена для збереження даних про всі можливі товари, які може отримати студент в обмін на накопичені бонусні бали за успіхи протягом навчання на курсах. Вона містить наступні поля: назва товару, його короткий опис, зображення, посилання на обирання та закріплення товару за собою, ціна у відповідних бонусних балах та сервісні параметри, такі як дата створення, додавання та оновлення товару.

Таблиця *lc_lessons* необхідна для збереження інформації про всі створенні заняття на відповідному із курсів. Вона містить наступні поля: назва заняття, дата, ФІО викладача та список студентів. Процес формування обліку реалізовано таким чином, що викладач на кожне нове заняття створює новий урок, заповнюючи вище вказані параметри та виставляє оцінки. В свою чергу для збереження оцінок студентів розроблена таблиця *lc_scores* яка містить поля для зберігання унікального ідентифікатора студента, викладача, уроку та відповідну оцінку.

Таблиця для забезпечення повноцінного функціонування розділу навчальних матеріалів *lc_root_directories* призначена для зберігання інформації про розміщення файлів та каталогів для кожного навчального курсу. Вона містить поля: напряму, курсу, ідентифікатор викладача та шлях до відповідної директорії. Наступна реалізація дозволяє легко регламентувати до яких директорій користувач має доступ в залежності від відвідуваних курсів, а також до яких папок та файлів має доступ відповідний викладач.

3.2.2 Створення індексів БД

Перш за все, при створенні індексів БД необхідно проаналізувати всі запити та виділити серед них ті які можуть використовуватися найчастіше за рядом причин.

Найчастіше використовуваним запитом є запит електронної пошти при авторизації в системі тому, що кожен користувач повинен пройти цей процес. Беручи до уваги той факт, що користувачів в системі може бути велика кількість, а також при одночасному запиті на авторизацію великої кількості користувачів може значно збільшитися час виконання таких запитів, що негативно вплине на швидкодію додатку. Тому вирішено створити індекс в для поля *email* у таблиці *lc_users* (рис. 3.5).

```
$table->string('email', 100)->unique();
```

Рисунок 3.5 – Приклад створення унікального індексу БД

Після проходження авторизації користувач потрапляє на головну сторінку з блоком новин. Беручи до уваги, що таких новинних блоків на сторінці не більше 10-ти, то застосування для запитів виводу блоку новин не є доцільним. Сторінка навчальних матеріалів містить також не більше 10-ти навчальних курсів, тому в даному випадку також не є доцільним застосування унікальних індексів, так як це не вплине на швидкість кінцевого результату виконання запиту з пошуку необхідного матеріалу. Зважаючи на те, що студентів досить велика кількість, тому реалізацію пошуку за ім’я або прізвищем на сторінці користувачів використання унікального індексу є доцільним. Тому вирішено створити індекси в таблиці *lc_users* для поля *name* та *surname* відповідно, що повинно значно пришвидшити пошук необхідного користувача. Використання індексів при пошуку за іншими параметрами, такими як філіал або курс не є доцільним. На сторінці філіалів навчального закладу, кількість яких зазвичай не більше 5-ти також не є доцільним використання індексів тому, що це не призведе до

пришвидшення виконання запиту при його пошуку. Навчальні групи, кількість яких може бути досить великою потребують унікального індексу при фільтрації по назві. Тому вирішено створити для таблиці *lc_groups* унікальний індекс поля *name*, що пришвидшить пошук за персональною названою групи. Фільтрація груп за курсом або напрямом не потребує застосування індексів при пошуку, внаслідок незначної кількості відповідно курсів та напрямків.

На сторінці відображення успішності студентів, а саме відображення переліку занять для конкретної групи не потребує створення унікального індексу внаслідок того, що зазвичай кількість занять одної групи не перевищує відмутку в 25. Тому застосування в даному випадку індексу ніяким чином не відобразиться на швидкодії пошуку занять певної групи.

3.3 Розробка моделей БД за допомогою ORM

Беручи до уваги те, що база даних персонального кабінету складається з 14-ти таблиць, то необхідно розробити 12 моделей для роботи з БД. Оскільки найважливішою таблицею даної бази даних є таблиця користувачів, яка зберігає інформацію про всіх користувачів персонального кабінету, а саме студентів, викладачів та адміністраторів.

Приклад програмної реалізації моделі для таблиці користувачів зображено на рис. 3.6. В моделях подібного типу існує зазвичай 4 базові властивості, такі як *cast, hidden, fillable* та *table*. Кожна модель таблиць мають всі необхідні методи для повноцінної роботи з базою даних. В свою чергу властивість *fillable* призначена для отримання від інтерфейсу асоціативного масиву даних, у яких значення ключа буде відповідати певному полю та його значенню. За допомогою такого методу можливо набагато простіше створювати нові моделі БД. Все що необхідно, так це передати асоціативний масив у конструктор відповідної моделі і

на виході отримати повнофункціональний клас у якому передані поля є його властивостями, а відповідно передані значення – значеннями цих полів.

```
class User extends Authenticatable
{
    protected $fillable = ['name','surname','telephone','email','password','type',
        'course_id','reset_code','reseted','avatar'];
    protected static $statuses = [
        'students' =>['entrant','student','graduate','specialist',],
        'teachers' => ['teacher_main','teacher_reserve',]
    ];
    public function usersData()
    {
        return $this->hasMany(UserData::class );
    }
    public function courses()
    {
        return $this->belongsToMany( Course::class, 'lc_users_courses' );
    }
    public function groups()
    {
        return $this->belongsToMany( Group::class, 'lc_users_groups' );
    }
    public static function getAllStudents()
    {
        return self::select('lc_users.id','lc_users.name','lc_users.surname')
            ->orderBy('lc_users.created_at', 'DESC')
            ->get();
    }
```

Рисунок 3.6 – Модель БД для таблиці користувачів

Властивість *cast* призначена для приведення даних будь-якого поля до певного типу, наприклад цілочесельного. Застосування даної властивості відбувається наступним чином: надходячи до бази даних кортеж даних перед збереженням перевіряється на приведення типів, якщо такі перетворення необхідні то відповідні значення проходять процес приведення. В даному випадку є необхідність приведення полів *id, course_id* та *rating* до значення цілого числа.

Властивість *table* призначена для чіткого зазначення яка модель до якої таблиці відноситься. За допомогою вбудованих властивостей фреймворку є можливість автоматично зв’язувати модель з відповідною їй таблицею за допомогою визначення моделі такої ж назва як і у таблиці але у однині. Наприклад якщо таблиця має назву *users*, то відповідна їй модель буде мати назву *User*.

За допомогою властивості *hidden* досягається захист бази даних від запису некоректних даних. Дана методика здійснюється за допомогою визначення які поля таблиці не можливо заповнювати просто передавши їх звичайним чином за допомогою властивості *hidden*.

Також в моделях таблиць бази даних вказуються всі їхні зв'язки з іншими таблицями за допомогою спеціальних методів. Ці методи дозволяють зручно витягувати необхідні дані із пов'язаних таблиць. Таким чином проаналізувавши модель таблиці *lc_users*, можна зробити висновок що вона має зв'язки із таблицями *lc_courses* та *lc_users_data*, за що відповідають відповідні методи. При виклику даних методів відбувається запит до цих таблиць для отримання певних даних. Таким чином моделі таблиць дозволяють працювати із базою даних, витягувати або зчитувати з неї дані без використання синтаксису SQL. Моделі виконують роль прошарку між базою даних та веб-додатком.

3.4 Розробка RESTfull API

Наступним кроком розробки персонального кабінету студента – це розробка REST API (рис. 3.7). Беручи до уваги, що дана система закритого типу, тобто для отримання доступу до неї будь-якого користувача необхідно пройти процес авторизації та отримати веб-токен для подальших дій. Після авторизації користувача, в процесі його переміщення по сторінкам додатку, перед тим як перейти на відповідну сторінку вбудований інтерфейс *middleware* буде перевіряти чи має користувач права на її відвідування. У випадку достатніх прав користувачу буде відображена необхідна сторінка, а у випадку недостатності він буде перенаправлений на сервісну сторінку із повідомленням про недостатність його прав у системі. Типова архітектура HTTP шляхів у веб-додатках зображена на рис. 3.7.

```
/**
 * users routes
 */
Route::post('/users/filter/{name}', 'UserController@filterStudents')
    ->where(['name' => '[A-Za-z]+'])->middleware('admin');
Route::post('/users/filterbysurname/{sname}', 'UserController@getStudentsBySurname')
    ->where(['name' => '[A-Za-z]+'])->middleware('admin');
Route::post('/users/delete/{id}','UserController@destroy')->middleware('admin');
/**
 * groups routes
 */
Route::group(['middleware' => ['auth','admin'], 'prefix' => 'groups'], function (){

    Route::get('/', 'GroupController@index')->name('groups');

    Route::get('/data',  'GroupController@getGroupsData');

    Route::post('/add', 'GroupController@store');

    Route::put('/update/{groupId}', 'GroupController@update')
        ->where(['groupId' => '[0-9]*']);

    Route::post('/filter', 'GroupController@filterGroups');

    Route::delete('/delete/{id}','GroupController@destroy');

    Route::post('/get-teacher-by-course-and-direction','GroupController@getTeacher');

    Route::post('{id}/restore','GroupController@restore');
});
```

Рисунок 3.7 – REST API для персонального кабінету

Як можна побачити, для функціонування такого механізму необхідно передати до *middleware* асоціативний масив, у якому ключем буде назва необхідного веб-компоненту, а значенням – метод за допомогою якого буде перевірятися чи може даний користувач відвідати певну сторінку або відправити або отримати необхідні дані. Далі передається параметр *prefix*, який забезпечує автоматичне додання префіксу до всіх шляхів. В даному випадку такий префікс вказано як *users*.

Кожен шлях навігації у веб-системі складається із певних частин, кожна з яких відповідає за те, який контролер фреймворку буде опрацьовувати даний шлях. Кожне правило для певного шляху починається із зарезервованого слова

Route, далі вказується який метод відправки запиту на сервер буде використовувати. Таких запитів до сервера існує п’ять основних типів (табл. 3.1).

Таблиця 3.1 – HTTP методи відправки запитів

Метод запиту	Результат виконання
POST	Відправка персональних даних на сервер
GET	Отримання даних із сервера
PUT	Оновлення або заміна даних
PATCH	Оновлення або заміна полів для таблиці
DELETE	Видалення даних з БД

Після того як визначено необхідний метод відправки запиту, формується URL адреса та виклик певного контролера фреймворку для обробки даного шляху. Далі передається власне метод обраного контролера, який буде опрацьовувати даний шлях, що отриманий із адресного рядка браузера. За допомогою додаткового методу *name* буде динамічно формуватися адреса певного шляху у адресному рядку в залежності від переданого параметру.

Використання ключового слова *resource* передбачає наявність 7-ми вбудованих у фреймворк методів для роботи із даними веб-додатку. Ці вбудовані методи є однаковими для всіх контролерів додатку, а також описують всі можливі дії із переданим їм даними.

Для роботи із користувачами розроблена наступна структура шляхів у веб-додатку (табл. 3.2). Дані методи необхідні для додавання, редагування, оновлення та видалення інформації про користувачів персонального кабінету, а також відображення веб-сторінки.

Таблиця 3.2 – Структура шляхів та методи їх обробки

Метод запиту	Адреса	Метод обробки	Результат виконання
GET	users/	*IndexPage*	Відображення шаблону сторінки
GET	users/getData	*GetUsersData*	Отримання інформації про всіх користувачів
POST	users/add	*StroreUser*	Створення нового користувача
PUT/PATCH	users/updated/{userId}	*updateUser*	Редагування користувачів
POST	/ users /filter	*filterUsers*	Фільтрація користувачів
DELETE	users/delete/{ userId }	*DestroyUser*	Видалення певного користувача
POST	users /{userId} /restore	*RestoreUser*	Відновлення видаленого користувача

Перелік наведених вище методів покриває всі потреби для повноцінної роботи із персональною інформацією користувачів системи. Розроблені методи та шляхів для додавання нових користувачів, оновлення їх персональних даних, видалення, відновлення раніше видалених користувачів, а також отримання повної інформації про них. Із додаткових методів створені фільтрації користувачів за таким параметрами як: ім’я, прізвище, напрям та курс навчання, а також філіал навчального закладу.

3.5 Розробка серверної логіки основних модулів

На головній сторінці персонального кабінету розміщені блоки із індивідуальними та актуальними новинами навчального закладу. За відображення даного блоку із новинами відповідає відповідно контролер фреймворку – *TeaserController* (рис. 3.8).

```
class TeaserController extends Controller
{
    public function getTeasers()
    {
        $greetings = $this->getTeaserBuilder( take: 1, type: 'greetings');
        $promo = $this->getTeaserBuilder( take: 1, type: 'promo');
        $articles = $this->getTeaserBuilder( take: 2, type: 'article');
        return response([
            'greetings' => $greetings,
            'promo' => $promo,
            'articles' => $articles,
            'can_edit' => Auth::user()->type === 'admin',
        ]);
    }

    public function create(TeasersRequest $request){
        $teaser = new Teaser();
        $data = $request->all();
        if($request->hasFile('image'))
            $data['image'] = parent::saveImage($request->file('image'), previousFile: Teaser::DEFAULT_IMAGE,
                defaultImage: Teaser::DEFAULT_IMAGE, storagePath: 'teasers');
        return response($teaser->map($data));
    }

    public function edit(Request $request)
    {
        $teaser = Teaser::findOrFail($request->id);
        $data = $request->all();
        unset($data['image']);
        if($request->hasFile('image')) {
            $data['image'] = parent::saveImage($request->file('image'), $teaser->image,
                defaultImage: Teaser::DEFAULT_IMAGE,  storagePath: 'teasers');
        }
        return response($teaser->map($data));
    }

    public function destroy(Request $request)
    {
        Teaser::findOrFail($request->id)->delete();
    }
```

Рисунок 3.8 – Контролер обробки головної сторінки системи

Після авторизації, користувач потрапляє на головну сторінку системи, за відображення якої відповідає контролер *TeaserController*, а саме його метод *getTeasers*, який виконує запит до БД та повертає json об’єкт із переліком блоків новин із детальною інформацією. Даний метод збирає всі блоки новин, які є актуальними для конкретного користувача, формуючи з них асоціативний масив.

Даний масив містить у ролі ключів – назву новинних блоків, а в ролі значень – об’єкт із повною інформацією про відповідну новину. Слід зазначити, що в кінці даного асоціативного масиву розміщено поле із вказанім типом користувача. Це поле необхідне для того, щоб на клієнтській частини можна визначити тип користувача, а також на основі цієї інформації відображати чи ні блок із можливістю редагування або видалення даного блоку новин.

Також даний контролер містить наступні методи: *create*, *edit* та *destroy*. Метод *create* призначений для створення нової новини. Даний процес розпочинається із того, що на сервер надходить об’єкт із усіма даними для новини, а саме назва, коротка інформація, зображення та посилання на повний текст новини. Далі відбувається перевірка та валідація переданих параметрів, і при успішному завершенні відбувається занесення даних до бази даних. Якщо валідація завершена із помилкою, то на клієнтську частину надходить повідомлення про помилку. У відповідь про успішне занесення даних до бази даних на клієнтську частину відсилається запит із статусом 200 та новим об’єктом який містить всі новини, враховуючи щойно створену.

В свою чергу метод *edit* необхідний для оновлення вже існуючих новин. Спочатку відбувається відсилання запиту із клієнтської частини, який містить об’єкт із оновленими даними та унікальним ідентифікатор тієї новини, що редагується. Далі на серверній частині відбувається валідація отриманих даних і у випадку успішного завершення цього процесу відбувається формування запиту до БД для пошук новини за отриманим ідентифікатором. Якщо валідація завершена із помилкою, то на клієнтську частину надходить повідомлення про помилку. При успішному пошуку новини за ідентифікатором відбувається оновлення даних цієї новини та запис оновленої інформації до бази даних. У відповідь на клієнтську частину надходить запит із статусом 200 та об’єктом, який містить оновлений перелік новин із новинного блоку.

Метод *destroy* призначений для видалення новини за її унікальним ідентифікатором, який надходить із клієнтської частини додатку. У випадку, коли такий ідентифікатор новини знайдений у базі даних, то відбувається видалення новини та відправка статусу 200 на клієнт, про успішне видалення. У випадку коли такий ідентифікатор не знайдений, то на клієнт відправляється особливий статус та повідомлення про помилку.

Аналогічним чином розроблені всі інші контролери для реалізації навчальних груп, навчальних матеріалів тощо. Всі вони базуються на вище описаних трьох методах і можуть бути опціонально доповнені специфічними методами для реалізації певних функціональних можливостей.

Оскільки вся система персонального кабінету побудована навколо користувача, то слід розглянути процес створення нового користувача (рис. 3.10).

```
$status = 200;
$statuses = User::getUserStatuses();
try {
    if( in_array($request->type, $statuses[ self::STUDENTS_KEY ] ) ) {
        $userType = self::STUDENTS_KEY;
    } else {
        $userType = self::TEACHERS_KEY;
    }
    $user = new User([
        'name' => $request->name,
        'surname' => $request->surname,
        'email' => $request->email,
        'telephone'=>$request->telephone,
        'password' => bcrypt('12345'),
        'type' => $request->type,
    ]);
    $user->save();

    $user->courses()->attach($request->course);
```

Рисунок 3.10 – Реалізація можливості додання нового користувача

Спочатку проводиться перевірка отриманих даних із клієнтської частини та подальша їх валідація. У випадку успішного завершення процесу перевірки отриманих даних відбувається створення нової моделі та заповнення її щойно отриманими даними. Лише після цього отримані дані записуються в базу даних та

реалізація зв’язків між студентом та обраним курсом для проходження навчання. Далі проходить процес запису додаткових даних отриманих від користувача у допоміжну таблицю. І після цього відбувається підрахунок загальної кількості користувачів з урахуванням ново створеного та повернення json об’єкту на клієнтську частину разом зі статусом відповіді 200 – в разі успішного завершення додавання нового користувача. Об’єкт відповіді містить масив який містить поле зі значенням загальної кількості користувачів відповідного типу, масиву із переліком усіх користувачів із їхніми персональними даними.

У разі отримання помилки в процесі створення нового користувача у випадку некоректності даних або якщо переданий номер телефону або адреса електронної пошти вже зареєстрована у системі – на клієнтську частину повертається відповідь зі статусом 422, який містить код помилки та текст повідомлення.

3.6 Розробка клієнтської логіки основних модулів

Розробка модулів клієнтської частини завжди розпочинається із поділу всього веб-додатку на дрібні компоненти, кожен з яких виконує свою певну роль. Чим детальніше розділено інтерфейс на компоненти тим зручніше в подальшому підтримувати такий код, додавати нові функціональні можливості, а також не перенавантажувати компоненти не властивими для них можливостями. Зазвичай веб-компоненти поділяються на функціональні компоненти та компоненти-класи. Функціональні компоненти зазвичай використовують для реалізації тих модулів системи в яких не потрібно звертатися до серверу, обробляти будь-які користувацькі події тощо. Щодо компонентів класів, то вони використовуються у тих випадках, коли безпосередньо в самому компоненті відбувається обробка подій, звернення до серверу додатку тощо.

В реалізованому таким чином веб-додатку всі компоненти складають деревовидну ієрархію та працюють за допомогою технології віртуального DOM. Дана технологія розроблена спеціально для цього фреймворку і працює наступним чином: при завантажені веб-сторінки, створюється копія DOM дерева, яка і називається віртуальний DOM. Далі при будь-яких змінах у компонентах веб-додатку, віртуальне дерево порівнюється із реальним і визначається який компонент отримав зміни, і після цього виконується перерендеринг лише цього компонента, а не всієї сторінки. За допомогою даної технології вдається досягти великої швидкодії вед-додатків.

Синтаксис фреймворку React, а саме JSX нагадує синтаксис розмітки веб-документу HTML, але з деякими специфічними особливостями та новими можливостями (рис. 3.11).

```
export default class App extends Component {
  render() {
    return (
      <App>
        <Header />
        <Nav />
        <Content />
        <Footer />
      </App>
    );
  }
}
```

Рисунок 3.11 – Синтаксис JavaScript фреймворку React

В даному випадку, за допомогою функції *render* відбувається рендеринг веб-додатку, який складається із шапки, меню, області для основного контенту та футеру. Кожен з цих компонентів складається опціонально із декількох дрібніших компонентів. Для прикладу головна сторінка персонального кабінету студента містить наступні глобальні компоненти: шапка, головне меню, поле для основного контенту та футер. В свою чергу найважливішим компонентом серед них є

компонент основного контенту, так як він складається із низки інших компонентів. Такі компоненти як шапка, меню та футер виконують лише роль відображення переданих у них даних із глобального компоненту *App*. Щодо компоненту який відповідає за відображення основного контенту сторінки, то він складається із компонентів заголовку сторінки, блоку новин, блоку сторінкової навігації. Блок новин в свою чергу складається із компонентів які відповідають за відображення певної новини. Кінцевий в ієрархії даної сторінки компонент новини містить у собі спеціальну JSX розмітку його структури, а саме блоки відображення заголовку новини, зображення, короткого опису та посилання на повний текст новини, та логіки при натисненні на кнопки редагування новини тощо.

Розглянувши компонент відображення шапки сторінки, варто вказати ту особливість, що даний компонент є єдиним для всіх сторінок персонального кабінету (рис. 3.12). Тому виділення його в окремий компонент вирішує завдання, яке полягає в редагування певної інформації даного компоненту, наприклад номера телефону тощо. При такій реалізації досить змінити необхідну інформацію водному місці, безпосередньо у файлі шапки системи і ці зміни будуть застосовані для всіх сторінок де використовується даний елемент.

Такий синтаксис нагадує використання шаблону документа. В даному випадку за допомогою глобального об’єкту властивостей this.props, реалізована можливість отримання необхідний даних, таких як *userName, userStatus* та *userBonus* із глобального компоненту App. В свою чергу головний компонент у момент завантаження сторінки отримує всі необхідні даній із веб-серверу і записує їх у глобальний об’єкт станів. За допомогою станів реалізована можливість збереження даних веб-додатку для зручного використання їх із будь-якого місця системи.

```
export default class Header extends Component {
  render() {
    return (
      <header className='container'>
        <div className='row'>
          <div className='col-xs-12 col-md-3 col-lg-3 logoBlock'>
            <a className='logo' href='https://kurspc.com.ua'>
              <img src={logoImg} />
            </a>
          </div>
          <div className='col-xs-12 col-md-3 col-lg-3 userBlock'>
            <div className='userName'>Добро пожаловать, {this.props.userName}</div>
            <div className='status'>
              статус:<span> {this.props.userStatus}</span>
            </div>
            <div className='status'>
              <a href='#'>
                Твой бонус:
                <span>
                  <i className='fas fa-star'></i>{this.props.userBonus}
                </span>
              </a>
            </div>
          </div>
```

Рисунок 3.12 – Логіка реалізації шапки персонального кабінету

Щодо використання «розумних» компонентів при розробці користувацького інтерфейсу, то можна зазначити, що їх використання необхідне при необхідності передачі даних від головного компонента до його нащадка безпосередньо, а не через всіх нащадків які знаходяться на рівні вище за нього. Такий підхід при проектуванні забезпечить прозорість та зручність використання даних отриманих від головного компонента, та подальше тестування таких компонентів. За допомогою таких компонентів, які ще називають контейнерами досягається уніфікація глобального сховища даних для всіх компонентів та додатку в цілому, а також можливість отримувати та передавати дані за допомогою передачі необхідних методів у компонент. В результаті цього в деяких випадках присутня можливість відмовитися від створення локального сховища даних для конкретного компонента, а також значно більш просте використання переданих даних та відмова від використання функцій зворотного виклику при обробці подій. Для забезпечення такого процесу необхідне використання спеціальної

функції connect із екосистеми Redux. На її вхід подається дві функції *mapStateToProps* та *mapDispatchToProps*. Перша функція необхідна для отримання необхідних даних із глобального сховища даних і завантаження його в об'єкт *this.props*. Інша ж функція призначена для отримання необхідних методів та запис їх в об'єкт this.props для подальшого використання при відправці певних даних у глобальне сховище.

Приклад використання та синтаксис контейнерів зображено рис. 3.13.

```
class ToDo extends Component {
  render() {
    return (
      <div>
        <ToDoInput/>
          <ToDoList onDelete={this.deleteTask} taskList={taskList} />
          <Footer amount={taskList.length} activeFilter={activeFilter} />
      </div>
    );}}
const mapStateToProps = store => {
  console.log(store);
  return {
    tasks: store.tasks
  };
};
const mapDispatchToProps = dispatch => ({
  addTask: (id, text, isCompleted) => dispatch(addTask(id, text, isCompleted)),
  deleteTask: id => dispatch(deleteTask(id))
});
export default connect(
  mapStateToProps,
  mapDispatchToProps
)(ToDo);
```

Рисунок 3.13 – Синтаксис використання «розумних» компонентів

Як можна побачити із синтаксису використання redux контейнерів для використання даних із глобального сховища даних передаються дві функції які відповідають за отримання даних та обробку даних для передачі в глобальне сховище. Також для доступу до отриманих даних із компонента реалізовано обернення цього компонента в компонент вищого порядку, який вертається із функції *connect*.

3.7 Інтеграція системи кешування Redis

За допомогою подібних систем реалізується підвищення швидкості пошуку необхідних даних. Підвищення швидкості досягається за рахунок зберігання певних даних в оперативній пам’яті веб-сервера. Зазвичай дана система розміщується на окремому веб-сервері тому, що при перезавантаженні серверу додатку, всі дані із оперативної пам’яті знищуються. Зважаючи на раніше проведений аналіз, та визначивши що найбільш використовувана системою таблиця – є таблиця користувачів. Тому саме ця таблиця потребує застосування методів кешування.

Дані в кешованому вигляді зберігаються в системі Redis за методом ключ-значення. Тому при авторизації користувача системи, веб-сервер додатку отримує адресу електронної пошти та пароль. Здійснюється пошук електронної адреси за допомогою запиту до серверу кешування, і у випадку успішного пошуку результат відразу передається веб-серверу для подальшої обробки. В тому випадку, коли необхідної адреси не знайдено на сервері кешування, відбувається пошук у базі даних персонального кабінету. Пошук здійснюється за відповідним ключем, наприклад user.email. Після вдалого пошуку в базі даних, ця адреса електронної пошти записується в систему кешування та віддається веб-серверу для подальших необхідних дій. Таким чином завжди поповнюється база системи кешування, і в наступний раз пошук цієї адреси займе значно менше часу.

В свою чергу необхідно слідкувати за синхронізацією даних між базою даних додатку та базою даних системи кешування Redis. Тому, що може виникнути такий випадок, що користувач змінить якість свої персональні дані і оновлені дані запишуться в базу даних системи. А на момент пошуку в базі даних системи кешування будуть знаходитись застарілі дані і в результаті користувач отримає не достовірну інформацію. Тому необхідно налаштувати серверну частину таким чином, щоб при операціях із модифікування персональних даних

користувачів, після оновлення інформації в базі даних додатку, проводилось записування оновлених даних і в базу даних системи кешування Redis. В інакшому випадку буде відбуватися пошук в базі даних персонального кабінету, і ніякого збільшення швидкості роботи не буде.

3.8 Реалізація unit тестування основних модулів

Для перевірки модулів розробленої системи на наявність помилок, використовують unit тестування. Перш за все необхідно визначити необхідні модулі, які треба протестувати, а потім визначити вхідні дані та очікувані вихідні. Далі за допомогою бібліотеки для тестування *PHPUnit* відбувається обирання відповідного модуля для тестування, вказується вхідні дані для нього та ті дані які ми очкуємо отримати на виході. Далі запускається тестування, по завершенню якого отримується звіт із отриманими даними, та порівнянням отриманих та очікуваних значень. Даний вид тестування дуже зручний у використанні в тестовому режимі або після перенесення розробленого веб-додатку на робочий веб-сервер, так-як після процесу переносу виявляється безліч помилок по ряду причин.

Для прикладу розглянуто алгоритм підрахування бонусних балів користувача, в результаті чого розроблені наступні unit тести (рис. 3.14).

Кожен із цих методів має унікальну назву, для зручного виклику необхідного методу для тестування певного випадку. За допомогою використання методу бібліотеки тестування *assertEquals* здійснюється розрахунок бонусних балів користувача, в результаті якого виконується порівняння отриманих результатів із очікуваними переданими параметрами. В результаті коли ці параметри рівні – тест вважається успішно завершеним, про що свідчить відповідне системне повідомлення в консолі.

```php
class UserRaitingTest
{
    public function testingCommonStudentRaiting1()
    {
        $raiting = new Raiting();

        $this->assertEquals(30, $raiting->calculate(User::findOrfail(1)));
    }

    public function testingCommonStudentRaiting2()
    {
        $raiting = new Raiting();

        $this->assertEquals(20, $raiting->calculate(User::findOrfail(2)));
    }

    public function testingCommonStudentRaiting3()
    {
        $raiting = new Raiting();

        $this->assertEquals(80, $raiting->calculate(User::findOrfail(3)));
    }

    public function testingCommonStudentRaiting4()
    {
        $raiting = new Raiting();

        $this->assertEquals(50, $raiting->calculate(User::findOrfail(4)));
    }
}
```

Рисунок 3.14 – Методи для тестування підрахунку балів

Аналогічним чином і інші методи виконують подібну функцію, тільки для інших користувачів (рис. 3.15).

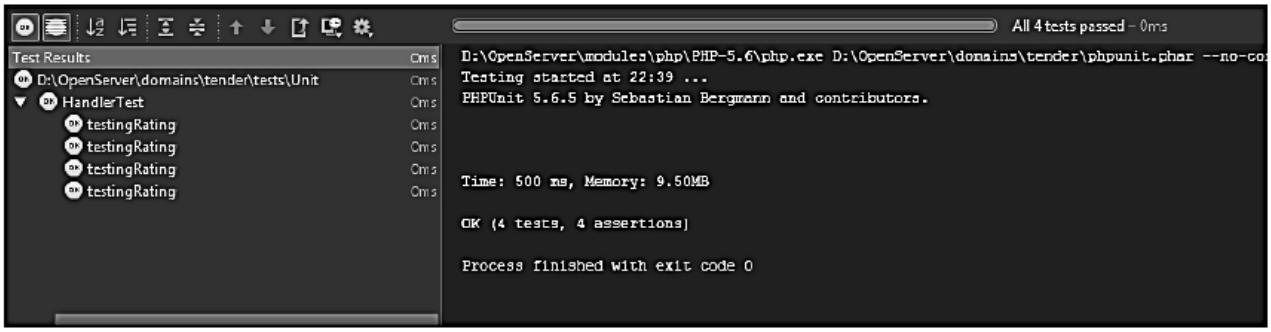


Рисунок 3.15 – Результат успішно завершення тестування

Як видно із зображення – всі тести виконалися успішно. Після процесу тестування в лівій частині доповнення для тестування відображається перелік усіх виконаних тестів, та додаткові параметри, а саме час за який цей тест виконався, об’єм затраченої пам’яті та кількість тестів.

В результаті не успішного завершення (рис. 3.16) тестування в лівій частині буде помічено спеціальними відмітками ті тести, в результаті запуску яких відбулися помилки. При натисненні на ці помітки в правій частині доповнення буде відображена інформація – з якої причини не виконався тест та будуть вказані очікувані результати та отримані в результаті проведення тесту.

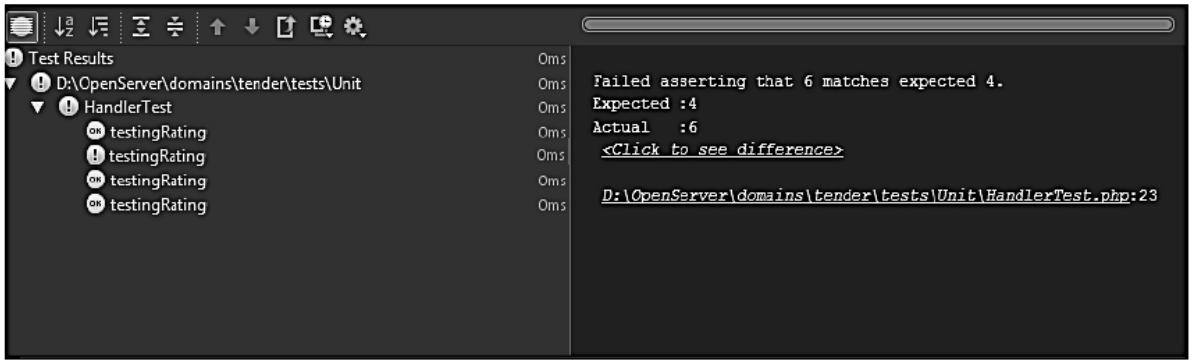


Рисунок 3.16 – Результат невдалого завершення тестування

За допомогою даного виду тестування протестовані всі необхідні модулі персонального кабінету користувача. Якщо в деяких модулях знайдено неточності або помилки, ці отримані дані проаналізовані та виправленні. Після виправлення помилок у відповідних модулях, проведено повторне тестування, задля виявлення додаткових помилок та пересвідчення у правильності роботи цього модуля або методу.

3.9 Масштабування веб-серверу на основі Amazon EC2

В наш час кількість користувачів збільшується великими кроками, та є дуже великою. Тому потужності одного веб-серверу для повноцінної роботи додатку недостатньо, тип більше для персонального кабінету студента курсів навчання робототехніці. Для таких цілей, тобто високонавантажених систем, якою можуть користуватися в один момент часу багато студентів необхідно масштабувати сервери. Такою системою – є система Amazon EC2, за допомогою якої можна просто та гнучко налаштувати процес масштабування серверів, та розподілити

навантаження між ними в моменти великих навантажень. Процес налаштування відбувається в два етапи: вибір типу сервера та його конфігурації. Використовуючи дану систему для масштабування сервері, адміністратор має можливість обрати тип серверу, а саме кількість оперативної пам’яті, кількість процесорі та їх характеристики, кількість місця на жорсткому диску та тип операційної системи (рис. 3.17).

Також присутня можливість обрання встановлення необхідного програмного забезпечення в залежності від типу операційної системи, такого як поштові клієнти, засоби SSL ліцензування, антивірусні програми та фаерволи тощо. Така реалізація дозволяє досить швидко обрати необхідний тип та конфігурацію серверу та створити новий сервер для свого веб-додатку. Після обрання конфігурації нового серверу та його створення, адміністратор має можливість переглянути та відредагувати певні його характеристики.

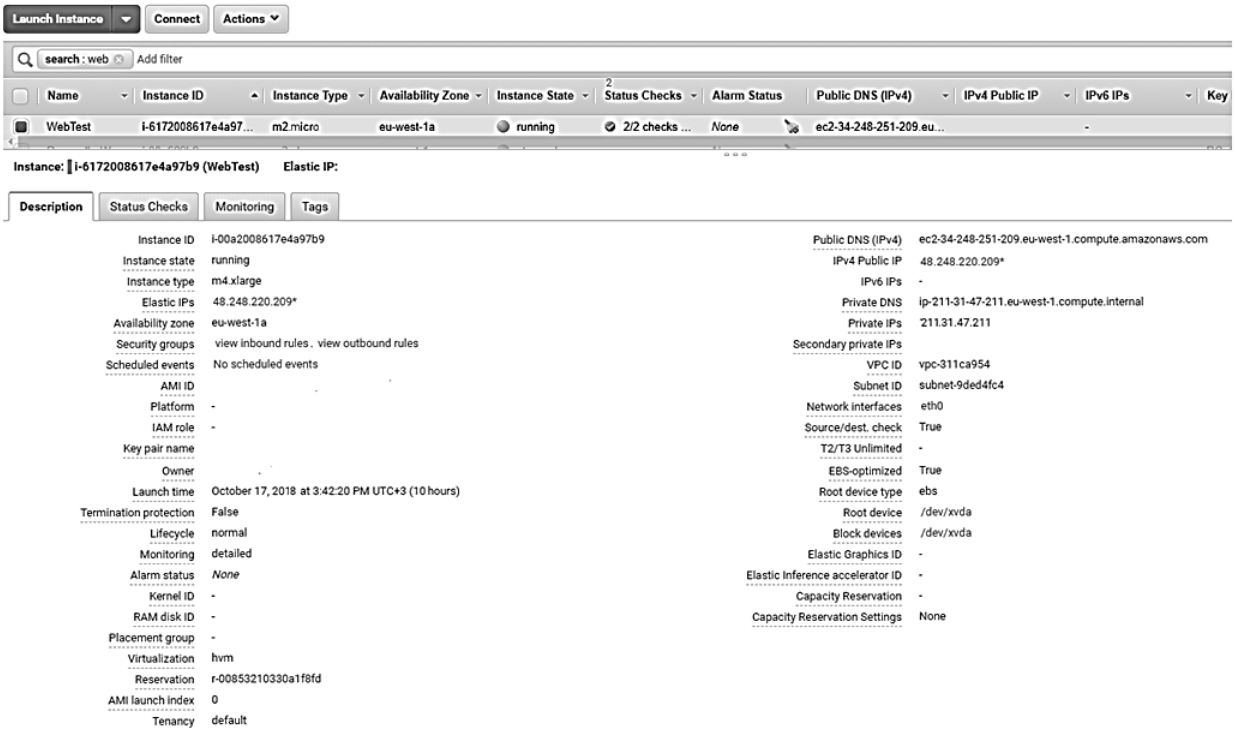


Рисунок 3.17 – Конфігурація створеного серверу за допомогою Amazon EC2

Після створення нового серверу необхідно встановити групу та правила безпеки для серверу. Даний набір правил містять в собі правила за якими визначається за яким протоколом та хто може здійснити підключення до серверу, швидкість підключення для певних веб-портів та сокетів, перелік дозволених та заборонених IP-адрес.

В нашому випадку в ролі операційної системи обрано UNIX-подібну систему *CentOS*, обраний один із запропонованих типів серверу, визначено адресу для підключення по зашифрованому протоколу SSH до серверу для проведення адміністративних функцій, налаштовано протокол захищений передачі даних HTTPS тощо. Зроблено копію налаштованого веб-серверу та додано його до групи масштабування, а також додано правило при якому у випадку перевищення навантаження на основний сервер більше ніж на 70% відбувається автоматичне налаштування та підняття додаткового серверу для забезпечення безперебійної роботи персонального кабінету.

Висновки до розділу

В даному розділі спроектовано структуру персонального кабінету студентів курсів навчання робототехніці. Визначено всі необхідні модулі, визначено логіку їх роботи та ті функціональні можливості котрі вони повинні вирішувати.

Обрано тип та спроектовано структуру бази даних для зберігання персональної та службової інформації. Розроблено моделі для більш зручної роботи бази даних та маніпулювання даними. Проаналізовано всі можливі запити до БД та визначено серед них ті, котрі використовуються набагато частіше і в результаті цього створено індекси для певних часто використовуваних полів таблиць.

Наступним кроком розроблено RESTfull API інтерфейс для забезпечення взаємодії клієнтської частини із сервером додатку. Розроблено основні модулі

серверної частини. Наступним кроком розроблявся інтерфейс клієнтської частини персонального кабінету. Поділено логіку роботи користувацького інтерфейсу на компоненти, кожен з яких виконує поставлену перед ним задачу. Далі налаштовано кешування часто використовуваних даних із бази даних за допомогою системи кешування Redis.

На останок створено та налаштовано масштабування серверів додатку, а також визначено набір правил при яких пікових навантаженнях буде підійматися додатковий сервер для забезпечення безперебійної роботи персонального кабінету.

РОЗДІЛ 4. ОПТИМІЗАЦІЯ ШВИДКОДІЇ ПЕРСОНАЛЬНОГО КАБІНЕТУ

4.1 Оптимізація серверної частини веб-додатку

4.1.1 Оптимізація роботи веб-серверу додатку

Веб-сервер – це найголовніша частина будь-якого веб-додатку. Він отримує запити від користувачів, формує відповідь та відправляє її до користувача. Нерідко формування відповіді займає досить тривалий час тому, що сервер може робити запити до бази даних веб-додатку, інших віддалених серверів та баз даних або проводити складні арифметичні та інші операції. В наслідок цього коли кількість запитів від користувачів в один момент часу зростає в соті або тисячі разів, то веб-сервер при неналежному налаштуванню або відсутній оптимізації може відмовити.

Існує декілька типових методів підвищення ефективності веб-сервера, до яких можна віднести:

– Стиснення запитів – веб-сервер стискає вміст запиту перед відправленням його на клієнт, а браузер, отримавши розгортає і відображає користувачу. За допомогою такого методу можна зекономити до 70% від розміру файлу із даними;

– Зменшення кількості запитів – кожен елемент на сторінці додатку, такий як зображення або скрипт – це новий запит до серверу. Чим менше користувач буде відправляти запитів на сервер тим краще, тому необхідно мінімізувати кількість зовнішніх запитів зі сторінки веб-додатку на сервер, а також використовувати клієнтське кешування, щоб зменшити кількість запитів від клієнта до веб-серверу;

– Налаштування ресурсів – веб-сервер без належного налаштування не може використовувати всі доступні йому ресурси у повному обсязі, тому необхідно в ручному режимі налаштувати необхідні сервісні параметри. До таких

параметрів можна віднести стиснення, логування, параметри обробки запитів, параметри обробки клієнтських підключень тощо.

Оптимізація будь-якої системи повинна розпочинатися з визначення характеристик системи та визначення пріоритетних параметрів оптимізації. Тобто визначення тих параметрів системи які необхідно оптимізувати.

Наступним кроком, після того як визначено оптимізаційні параметри необхідно проаналізувати систему та визначити її слабкі сторони. Ті вузькі місця системи, які можуть негативно вплинути на параметри оптимізації. Їх оптимізувати необхідно в першу чергу. В свою чергу може виявитися що вузьким місцем системи є не сам додаток або база даних, а інтерфейс взаємодії між ними. Дана робота бере за свою мету виявлення саме таких випадків, де вузьким місцем системи – є інтерфейс взаємодії бази даних та веб-серверу, та проведення оптимізації саме для таких випадків.

4.1.2 Критерії оптимізації

Критерієм оптимізації системи можуть бути різноманітні параметри. Головною задачею оптимізації є знаходження мінімум або максимум для цільової функції, тобто її екстремуму. У випадку, коли екстремум розташований зовні значень спеціальних параметрів, то у такому випадку в середині цієї області необхідно визначити найбільшу або найменшу величину цієї цільової функції. Основна мета оптимізації полягає у знаходженні компромісного рішення, у випадку коли декілька параметрів цільової функції можуть збільшувати значення критерію оптимізації, а інші – зменшувати. В даному випадку за такий параметр варто обрати час взаємодії клієнтської та серверної частин додатку. Таким чином за критерій оптимізації обираємо мінімізацію часу взаємодії клієнту і серверу.

У самому найпростішому випадку, при наявності одного запиту до серверу, процес взаємодії клієнту та серверу складається з наступних складових (рис. 4.1)

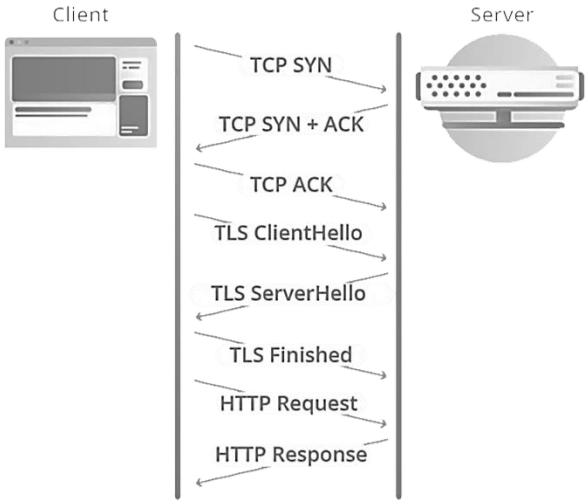


Рисунок 4.1 – Процес клієнт-серверної взаємодії

Загальний час обробки одного запиту ($T_{заг}$) до бази даних визначається наступним чином:

$$T_{заг} = T_{зд} + T_{пз} + T_{ком} + T_{озс} + T_{пр} + T_{орк}\,, \tag{4.1}$$

де $T_{зд}$ – час з'єднання с сервером, до складових якого можна віднести час створення об'єкту встановлення з'єднання, встановлення з'єднання, проведення автентифікацію користувача, встановлення спеціальних сервісних параметрів з'єднання, налаштування логіки для реакції на певні групи помилок.

$T_{пз}$ – час передачі даних між сервером та клієнтом, зазвичай має досить мале значення. Однак знехтувати цією складовою неможливо, так як можуть бути випадки коли результат запиту має менший обсяг ніж вміст самого запиту який передається з серверу.

$T_{ком}$ – час необхідний для компіляції запиту. Зазвичай передані запити поділяють на дві групи, а саме динамічні запити та запити збережених процедур. Запити з першої групи повинні бути скомпільовані при надходженні на сервер, також може бути вказано, що необхідно заново перезбирати запит при кожному новому виклику. Щодо другої групи, то це запити які вже попередньо

скомпільований і компіляція таких запитів не потрібна. Час компіляції складається з часу який необхідний на синтаксичний та лексичний аналіз запиту, час пошуку необхідного плану обробки в кеші, а також час самої компіляції запиту. Таким чином при наявності великої кількості таких запитів, то час компіляції буде істотно великим, тому оптимізація даного параметру може досить сильно зменшити цей час шляхом розділення запиту на більш прості.

$T_{озс}$ – час необхідний серверу для обробки запиту, тобто безпосередньо час необхідний для виконання запиту.

$T_{пр}$ – час необхідний для передачі запиту з сервера на клієнт. Даний параметр може стати критичним у випадку коли швидкість з'єднання досить низька, а обсяг даних для передачі великий. Такий час можна розрахувати наступним чином:

$$T_{пр} = \gamma C, \tag{4.2}$$

де γ – пропускна здатність каналу передачі; C – обсяг даних для передачі.

Дані для передачі можуть мати тип з фіксованою та динамічною довжиною. Також крім самих даних передається певна службова інформація, яка має різну довжину для своїх стовпців. Тому на основі цього отримуємо наступну формулу:

$$C = C_{здф} + C_{здд} + C_{сі}, \tag{4.3}$$

де $C_{здф}$ – загальний обсяг даних із фіксованою довжиною; $C_{здд}$ – загальний обсяг даних із динамічною довжиною; $C_{сі}$ – обсяг службової інформації.

Оскільки дані з фіксованою довжиною мають однакову довжину для різних кортежів в одних і тих стовпцях, то справедливо наступне:

$$C_{зфд} = N_{корт} \sum_{i=1}^{M_{фд}} C_i, \tag{4.4}$$

де $N_{корт}$ – число кортежів в результуючому наборі; $M_{дф}$ – число стовпців з фіксованою довжиною; C_i – перший стовпець фіксованої довжини.

Беручи до уваги той факт, що значення стовпців з динамічною довжиною мають різні значення, то справедливо наступне:

$$C_{одд} = \sum_{j=1}^{N_{корт}} \sum_{i=1}^{M_{дд}} C_{dij}, \quad (4.5)$$

де $M_{дд}$ – кількість стовпців динамічної довжини; C_{dij} – довжина першого кортежу із динамічною довжиною.

На основі проведених перетворень загальний обсяг даних, які передаються можна визначити за формулою:

$$C = N_{корт} \sum_{i=1}^{M_{фд}} C_i + \sum_{j=1}^{N_{корт}} \sum_{i=1}^{M_{дд}} C_{dij} \quad (4.6)$$

або

$$C = \sum_{i=1}^{M} \left(N_{корт} * C_i * f + (f-1) * \sum_{j=1}^{N_{корт}} C_{ij} \right) + C_{ci}, \quad (4.7)$$

де M – число стовпців в наборі даних; C_i – довжина першого стовпця; $C_ф$ – множина стовпців з фіксованою довжиною; $C_д$ – множина стовпців з динамічною довжиною.

При надходженні потоку запитів на серверну частину веб-додатку, на початку обробки здійснюється створення об’єктів для роботи з сервером, встановлюється з’єднання, а потім розпочинається робота з прийомо-передачі даних. Після закінчення роботи, а саме закриття з’єднання відбувається знищення цих сервісних об’єктів.

Слід зазначити, що час $T_{зд}$ буде кількісно визначений лише в перший раз, тобто при створенні з’єднання з сервером, для інших запитів цей час буде рівний 0, оскільки відсутня необхідність встановлення з’єднання. Зважаючи на те, що обробка різних запитів може тривати різну кількість часу, то за значення $T_{орк}$ необхідно прийняти $T_{\sum орк}$.

Щодо всіх інших запитів із потоку, то слід вважати, що вони виконуються більше одного разу, тоді буде вірним наступне твердження:

$$T_{заг} = T_{зд} + L\left(T_{пз} + T_{ком} + T_{озс} + T_{пр}\right) + T_{\sum орк}. \qquad (4.8)$$

4.2 Запити із зовнішніми ключами

В нашому веб-додатку досить часто використовуються запити із зовнішніми ключами, найкраще це можна побачити на реалізації функціоналу навчальних груп. Тобто такі запити використовуються у відношеннях з філіалами, курсами та студентами. Візуально каскадна таблиця відображена (табл. 4.1).

Таблиця 4.1 – Каскадна таблиця управління навчальними групами

Філіал	Курс	Група	Студент
Філіал 1	Курс 1	Група 1	Студент 1
			Студент 2
	Курс 2	Група 2	Студент 3
		Група 3	Студент 4
			Студент 5
Філіал 1	Курс 3	Група 4	Студент 6
		Група 5	Студент 7

В свою чергу каскадну таблицю можна описати у вигляді системи відношень.

$$\begin{cases} Col_2 = F(Col_1); \\ Col_3 = F(Col_2); \\ Col_N = F\left(Col_{N-1}\right). \end{cases} \qquad (4.9)$$

Такий спосіб запису відображає стовпець, який є аргументом, що він має більший рівень вкладеності ніж стовпець, з результатом залежності. При побудові

каскадної таблиці найчастіше використовують метод при якому створюють запит на вибірку до найбільшого рівня вкладеності, а потім для необхідних кортежів зробити запити до другого рівня вкладеності і так далі.

Для побудови повної та реальної каскадної таблиці запитів скоріше за все знадобиться велика кількість запитів, що значно вплине на параметр $T_{заг}$, в результаті чого він істотно зросте. Тому першочерговою задачею є зменшення кількості запитів, шляхом оптимізації.

Отримавши весь набір даних, необхідно починати їх обробляти, врахувавши наступні моменти:

– Процес обробки значно ускладниться, в результаті чого збільшиться $\sum T_{орк}$;

– Отриманий набір даних скоріше за все буде характеризуватися такою мірою як надмірність;

– Зменшиться час, який витрачався на компіляцію, лексичний та синтаксичний аналіз та виконання запитів у результаті перетворення отриманих запитів до одного типу.

SQL завжди намагається для кожного запису з першої таблиці підібрати всі записи з другої таблиці. Зазвичай ми утримуємо його від цього бажання задавши умову зв'язку двох таблиць. А ось коли спрацьовує OR *readers.speciality_id* IS NULL виходить, що умова істинно для будь-яких записів з другої таблиці.

Для таких випадків в SQL служить LEFT JOIN, який підбирає записи якщо вони є і повертає запис з першої таблиці навіть якщо в другій нічого немає.

Використавши вище описаний метод для системи персонального кабінету студентів навчання робототехніці, тобто проаналізувавши прості запити та створивши на їх основі складні запити, досягнуто позитивних результатів, які графічно зображені на діаграмі (рис. 4.2).

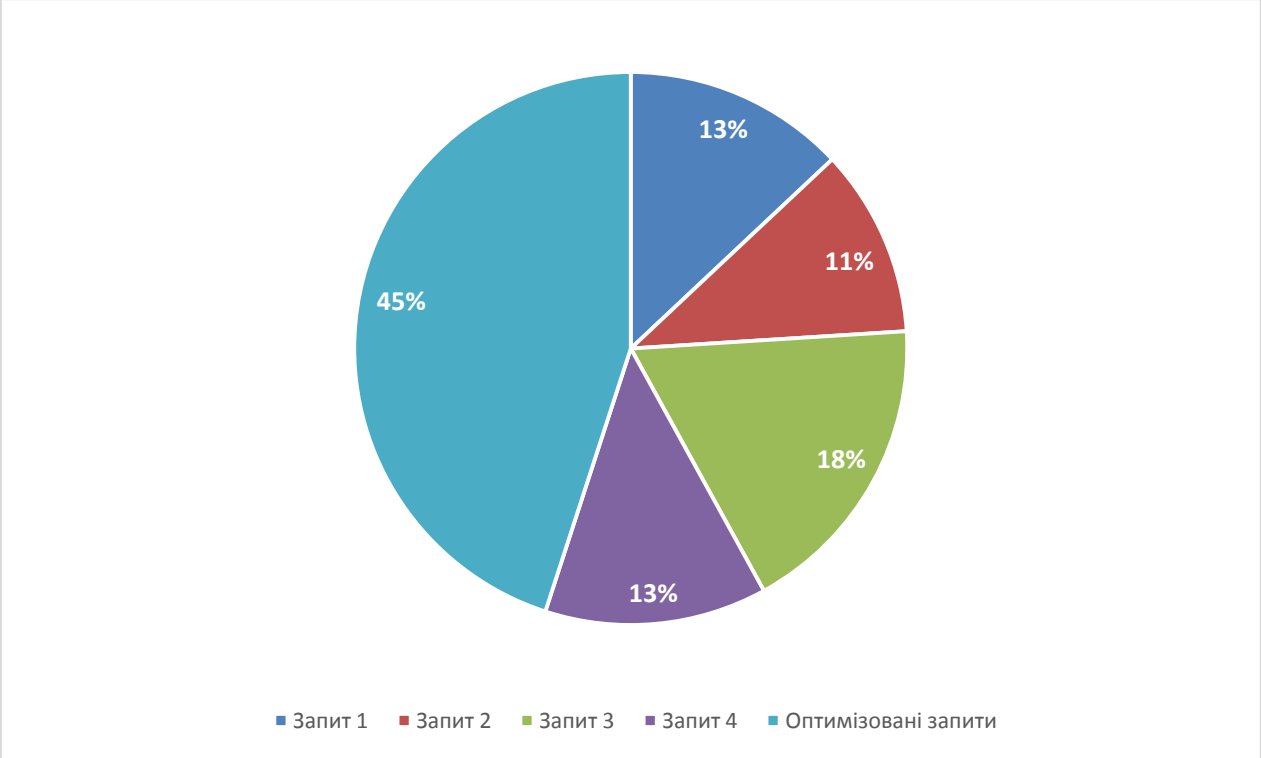


Рисунок 4.2 – Діаграма кінцевої оптимізації запитів

4.3 Метод кешування індексів

Даний метод оптимізації бази даних має дуже велику та безперечну перевагу, а саме його простота та можливість зберігання індексів в оперативній пам’яті. Щодо його недоліків, то можна відмітити те, що індекси які використовуються досить рідко зберігаються в оперативній пам’яті тривалий час, аж поки згодом їх не змістять більш свіжі індекси.

Слід описати алгоритм завантаження індексів в оперативну пам’ять СУБД:

– СУБД завантажує індекси в оперативну пам’ять для проведення статистики щодо кількості використань певних індексів за період ΔT;

– Після завершення процесу збору статистики СУБД завантажує в оперативну пам’ять лише найчастіше використовувані індекси;

– Якщо необхідний індекс відсутній в оперативній пам’яті, то у такому випадку його пошук виконується методом зчитування з фізичних накопичувачів БД;

– З плином деякого часу ΔT в оперативну пам’ять завантажуються тільки найчастіше використовувані індекси за умови якщо вони не завантажені туди раніше.

Зазвичай даний алгоритм реалізується в два етапи. Спочатку використовується звичайний алгоритм збору та визначення найчастіше використовуваних індексів за час ΔT. Завершальним етапом – є завантаження необхідних індексів в оперативну пам’ять СУБД за час ΔT, який можна розрахувати за допомогою формули:

$$\Delta T = \frac{K_i * T}{K}, \qquad (4.10)$$

де K_i – кількість індексів; K – кількість застосувань відповідних індексів; T – час виділений для збору статистики.

Скориставшись ідеєю даного методу та удосконаливши його доданням модулю, який реалізує процес збору статистики за допомогою серверної логіки та додання спеціальної сервісної таблиці у базі даних веб-додатку для накопичення статистики досягнуто наступного результату. В результаті реалізації даного методу отримано наступну інформацію для проведення аналізу: K_i = 16 індексів, K = 720 разів, T = 10 годин або 600 хвилин. Використавши дане відношення можна розрахувати час збору статистики індексів для нашої бази даних:

$$\Delta T = \frac{K_i * T}{K} \approx 13 \text{ хвилин}.$$

В результаті розрахунку, отримавши час збору статистики індексів, який приблизно рівний 13 хвилин, можна візуально відобразити використання індексів кожної години у вигляді діаграми (рис. 4.3).

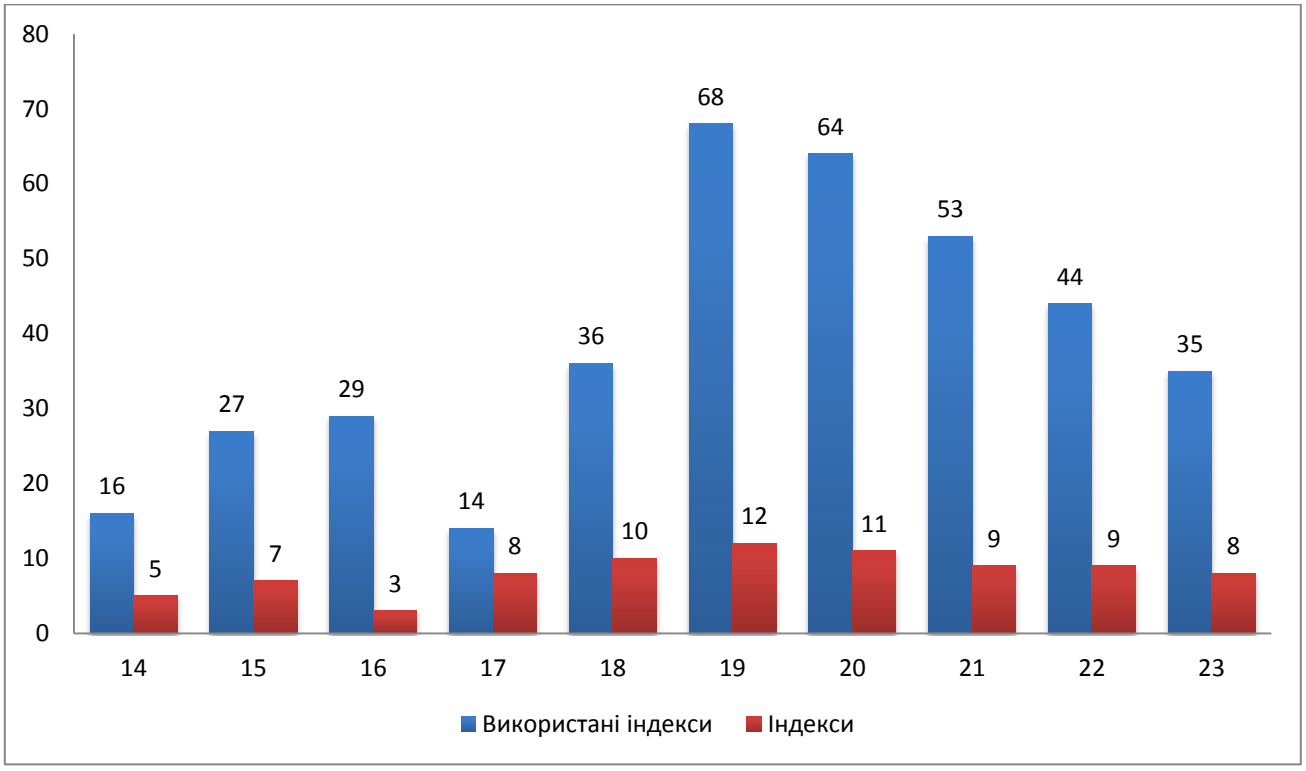


Рисунок 4.3 – Відношення кількості індексів та кількості запитів

Проаналізувавши отриману інформацію, на основі методу кешування індексів, найбільше навантаження на СУБД відбувається в проміжок між 19-ю та 21-ою годинами дня. Тому модифікувавши даний метод, а саме створення окремого додаткового кешу для тих індексів які використовуються у момент максимального навантаження системи, щоб вони завжди знаходилися в оперативній пам’яті. Така реалізація по-перше позбавить сервер від завантаження кешу по запиту, тобто пришвидшить виконання перших запитів, а по-друге завантажений таким чином індекс буде повністю відсортований, що додатково пришвидшить виконання запитів із використанням такого кешу. За допомогою даного методу вдалося оптимізувати швидкість пошуку близько 5%.

4.4 Оптимізація клієнтської частини

Оптимізація клієнтської частини полягає у зменшенні часу завантаження сторінки $T_{орк}$ на стороні користувача. На швидкість завантаження сторінки на стороні клієнта впливають наступні чинники:

– Оптимізація зображень – процес зменшення розміру зображення за рахунок стиснення зображень без помітних оку спотворень якості, а також використання сучасних типів зображень спеціально адаптованих для роботи у WEB;

– Мінімізація JavaScript, CSS та HTML – процес зменшення розміру відповідних файлів, шляхом прибирання зайвих пробілів, переносів рядка, а також можливе об’єднання декількох відповідних фалів JavaScript або CSS в один файл;

– Використання кешу браузера – процес переносу всіх статичних файлів на сервери CDN, які за допомогою своєї мережі серверів по всьому світу зберігають копії статичних файлів, коли користувач починає завантаження сторінки ці файли завантажуються із найближчого до клієнта серверу CDN, зменшуючи таким чином час завантаження;

– Переміщення блокуючих кодів – переміщення файлів які завантажуються досить тривалий час з блоку head в блок footer для того, щоб не блокувати завантаження контенту сторінки, коли сторінка з контентом завантажиться та всіма необхідними стилями, тільки після цього почнуть завантажуватися відповідні скрипти;

– Використання стиснення файлів – процес стиснення та відновлення файлів для зменшення їх розміру. Найпопулярнішим засобом для стиснення у WEB є технологія GZIP, яка працюючи на сервері стискає та відновлює файли в режимі реального часу;

– Оптимізація мобільного формату – процес оптимізації системи під переносні мобільні пристрої. Така оптимізація виконується за рахунок мінімізації

випадків асинхронного завантаження блоків коду, відмова від використання Flash блоків анімації, налаштування viewport для визначення ширини екрану мобільного пристрою, адаптація розміру елементів та шрифтів пропорційно відносно розміру екрану пристрою.

Зробивши оптимізацію клієнтської частини додатку, та проаналізувавши результати можна зробити висновки:

Головна сторінка системи (рис. 4.4), яка складається з блоку новин, меню та короткої інформації про користувача завантажується менше трьох секунд, а саме 2.65s, що є більш ніж достатнім.

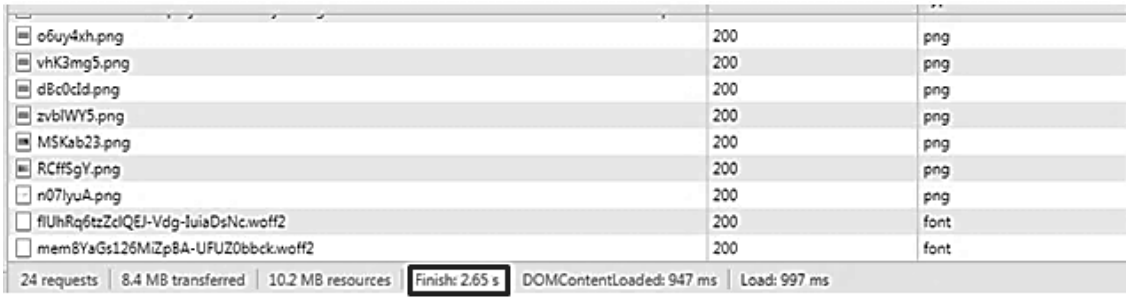


Рисунок 4.4 – Завантаження головної сторінки системи

Сторінка навчальних курсів (рис. 4.5), яка містить перелік курсів з короткою інформацією та зображенням завантажується трохи більше двох секунд, а саме 2.08s.

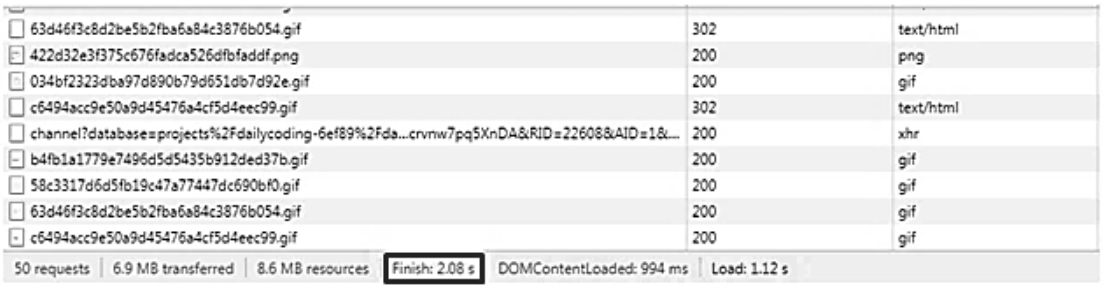


Рисунок 4.5 – Завантаження сторінки навчальних курсів

Сторінка з інформацією про навчальний курс (рис. 4.6), яка містить блоки з розширеною інформацією про курс, зображення та інше завантажується менше двох секунд, а точніше 1.65s.

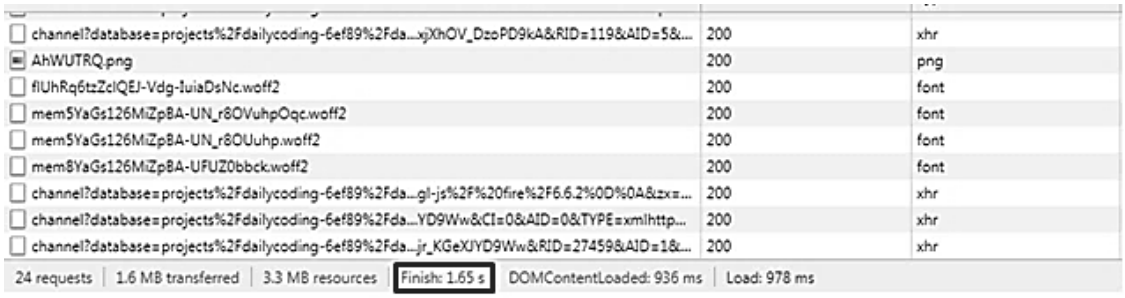


Рисунок 4.6 – Завантаження сторінки відповідного навчального курсу

Отже проаналізувавши результат оптимізації клієнтської частини персонального кабінету студента можна зробити висновок, що оптимізація покращила роботу додатку та підвищила швидкість його завантаження на стороні клієнта та відповідає усім нормам та стандартам Google PageSpeed Insights.

Висновки до розділу

В даному розділі проведено оптимізацію роботи персонального кабінету. За критерій оптимізації обрано швидкість завантаження сторінок. Проведено оптимізацію бази даних, а саме визначено прості запити які в подальшому об’єднано в більш складні. Також модернізовано метод кешування індексів БД, а саме створення окремого додаткового кешу для тих індексів які використовуються у момент максимального навантаження системи, щоб вони завжди знаходилися в оперативній пам’яті. Така реалізація по-перше позбавить сервер від завантаження кешу по запиту, тобто пришвидшить виконання перших запитів, а по-друге завантажений таким чином індекс буде повністю відсортований, що додатково пришвидшить виконання запитів із використанням такого кешу. Також проведена оптимізація серверної частини додатку, а саме виконано стиснення запитів, об’єднання декількох запитів в один та налаштування ресурсів веб-серверу. В останню чергу виконано оптимізацію клієнтської частини, а саме стиснення зображень, стиснення та об’єднання додаткових скриптів, а також використання кешу браузеру.

РОЗДІЛ 5. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

Необхідно провести маркетинговий аналіз стартап проекту, щоб визначити його потенціальні можливості провадження на ринку програмного забезпечення.

5.1 Опис ідеї проекту

Розробка сучасного, надійного персонального кабінету курсів навчання студентів робототехніці є головною ідеєю даного стартап проекту. Одним із основоположних факторів даного кабінету є забезпечення належної швидкості роботи та зручності для студентів. При дотримані даних факторів така система повинна витримати високі навантаження при одночасному користуванні великою кількістю студентів, та забезпечити зручне отримання адміністраторських функцій та навчання студентів.

Далі необхідно більш детально розглянути напрямки застосування, цільову аудиторію та переваги для користувачів даного персонального кабінету (табл. 5.1).

Таблиця 5.1 – Опис головної ідеї даного стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробити персональний кабінет для студентів курсів навчання робототехніці	Застосування у приватних навчальних закладах або гуртках для автоматизації навчального процесу	Приватні навчальні заклади отримують зручний програмний продукт за допомогою, якого автоматизується навчальний процес у їхньому навчальному закладі
	Застосування студентами для підвищення рівня якості освіти	Студенти даного навчального закладу отримують можливість дистанційно відвідувати навчальні курси переглядати навчальні матеріали обраних дисциплін та отримувати знання більш якісно

Наступним кроком необхідно визначити для даного персонального кабінету його техніко-економічні переваги перед вже існуючими аналогічними програмними продуктами. В якості аналогів нашого веб-додатку обрано систему навчання програмуванню GeekBrains та персональний кабінет онлайн курсів HTMLAcademy. Далі необхідно порівняти обрані аналогічні веб-системи із нашим програмним продуктом для визначення сильних слабких та нейтральних показників (табл. 5.2).

Таблиця 5.2 – Визначення слабких, нейтральних та сильних сторін нашого веб-додатку

№	Техніко-економічні характеристики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій веб-додаток	GeekBrains	HTML Academy			
1	Належний веб-дизайн системи	+	+	-		+	
2	Зручність використання системи	+	-	+		+	
3	Швидкість роботи системи	+	-	-			Має високу швидкість роботи
4	Функціональна можливість роботи із навчальними матеріалами	+	-	+		+	+
5	Створення навчальних груп, а також занять	+	-	-			Реалізована можливість доступу до навчальних матеріалів для студентів

Закінчення таблиці 5.2

6	Оцінювання студентів на заняттях	+	-	+		+	Можливість студента отримувати бали за активність на заняттях
7	Завантаженість сторінок надлишковою інформацією	-	-	+		+	
8	Забезпечення масштабованості системи	+	-	+		+	

Із проведеного аналізу можна зробити висновок, що дана ідея має вагомі переваги перед своїми конкурентами.

5.2 Технологічний аудит ідеї проекту

Наступним кроком необхідно провести аудит веб-технологій за допомогою яких можна створити дану систему, тобто реалізувати головну ідею даного стартап проекту.

Таблиця 5.3 – Технології реалізації ідеї стартап проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Розробка персонального кабінету студента курсів навчання робототехніці, а також забезпечення високої швидкості його роботи	Застосування серверної мови програмування PHP	Наявна. Існує декілька фреймворків для реалізації подібних ідей, які забезпечують належний рівень безпеки та швидкодії	Вільна
		Застосування мови програмування Java	Наявна. Існує лише один фреймворк для реалізації подібних ідей	Вільна
		Застосування мови програмування Python	Наявна. Існує лише один фреймворк для реалізації подібних ідей	Вільна
Обрана технологія реалізації проекту: мова програмування PHP.				

Зробивши аналіз отриманої інформації про технології, за допомогою який можна реалізувати головну ідею даного стартап проекту, а саме розробити персональний кабінет студента курсів навчання робототехніці. В результаті цього обрана мова серверного програмування PHP, яка має декілька основних фреймворків для використання, які забезпечують належний рівень безпеки розроблених за допомогою них веб-додатків, а також високий рівень швидкості роботи високонавантажених систем, до яких саме відносить даний програмний продукт. В якості одного із запропонованих фреймворків вирішено обрати Laravel тому, що він має зручні можливості для роботи із базою даних, сучасні алгоритми забезпечення безпеки веб-додатків, а також великий спектр готових рішень окремих модулів, які можливо використовувати для скорочення часу та підвищення продуктивності розробки.

5.3 Аналіз ринкових можливостей впровадження даного стартап проекту

Наступним кроком необхідно визначити ринкові можливості, тобто ті можливості які потрібно використати в процесі запуску проекту, а також визначити ринкові загрози, що можуть призупинити або перешкодити процесу запуску стартап проекту (табл. 5.4).

Таблиця 5.4 – Характеристика потенційного ринку для запуску стартап проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	8
2	Загальний обсяг продаж, грн./ум.од	2500 в рік
3	Динаміка ринку	Зростає
4	Наявність обмежень	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	100%

На основі проведеного аналізу ринкових можливостей та можливих загроз, можна зробити висновок, що ринок є привабливим для реалізації головної мети стартап проекту.

Далі необхідно визначити цільову аудиторію цього стартап проекту (табл. 5.5).

Таблиця 5.5. Характеристика потенційних клієнтів стартап–проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Наявність персонального кабінету для приватних навчальних закладів або курсів навчання студентів	Цільовою аудиторією є приватні навчальні заклади які навчають студентів як очно так і віддалено, а також ті які заохочені отримати зручний інструмент для адміністрування та підвищення якості освіти	Приватні навчальні заклади та курси навчання студентів на однаковому рівні зацікавлені у використанні додатку такого типу	Вимогами до споживачів є наявність належної швидкості інтернет з’єднання, а також базові навички при роботі із веб-додатками такого рівня складності

На основі проведеного аналізу цільової аудиторії, а також порівняння зацікавленості між різними цільовими групами, можна зробити висновок, що дана система є однаково привабливою для приватних навчальних закладів та курсів навчання студентів. Щодо повноцінних навчальних закладів, то дана система потребує доопрацювання, а саме додання додаткових модулів, що реалізують певні специфічні функціональні можливості.

Наступним кроком необхідно зробити аналіз ринкового середовища, а саме визначити фактори основних загроз, що можуть вплинути на продаж розробленого персонального кабінету (табл. 5.6).

Таблиця 5.6 – Основні фактори загроз продажу

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Зменшення швидкості роботи персонального кабінету	Даний персональний кабінет має високу швидкодію за рахунок, гарно продуманої архітектури та проведеної оптимізації, але з плином тривалого часу додаток буде мати досить велику кількість даних, як корисних так і тимчасових системних, що і може призвести до зменшення швидкості роботи кабінету.	Звернення до команди розробників даного веб-додатку із проханням перерозподілення користувацьких даних із однієї таблиці в декілька, а також перевірка системи на тимчасові системні файли із подальшим їх видаленням.
2	Можливість появи певних системних помилок в додатку	Даний веб-додаток має певні зв’язки між основними модулями системи, а також уніфіковані правила прийомо-передачі даних між ними.	Доповнення існуючої документації до програмного продукту із детальним поясненням про типи даних для передачі між модулями системи.

Далі необхідно розглянути основні фактори тих можливостей, які можуть вплинути на розроблений продукт під час виходу його на ринок (табл. 5.7).

Таблиця 5.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Автоматизація роботи приватного навчального закладу або курсів навчання	За допомогою даного додатку можливо значно підвищити рівень якості освіти, а також оптимізувати навчальний процес за рахунок зменшеного часу на формування навчальних груп, завантаження навчальних матеріалів, виставлення оцінок, а також виконання адміністративних функцій	Підвищення рівня знань викладачів та адміністраторів даного персонального кабінету
2	Процес групової розсилки електронних повідомлень	Функціональні можливості даного персонального кабінету дозволяють розсилати електронні повідомлення для груп студентів або для окремих користувачів	Підвищення рівня знань адміністраторів додатку з питання створення та налаштування групових розсилок електронних повідомлень

Наступним кроком необхідно виконати ступеневий аналіз конкуренції (табл. 5.8).

Таблиця 5.8 – Ступеневий аналіз конкуренції

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства(можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: чиста	На ринку безліч конкурентів, що розробляють веб-додатки схожого типу	Використання реклами програмного продукту, а також підтримання належного рівня швидкодії та безпеки
2	Рівень конкурентної боротьби: національний	Даний персональний кабінет адаптований для використання в навчальних закладах по всій території України	Підвищення надійності функціонування модулів системи, а додання нових можливостей
3	Галузева ознака: внутрішньогалузевий	Конкуренція в сфері освіти та адміністрування навчальних закладів	Підвищення якості та гнучкості роботи даної системи
4	Конкуренція за видами товарів: товарно-видова	Конкуренція між аналогічними персональними кабінетами	Впровадження тих функціональних можливостей які відсутні у конкурентів та гра на їх недоліках
5	Характер конкурентних переваг: цінова	Відношення вартості персонального кабінету до показників його якості та надійності	Продаж даної системи за співвідношенням ціна/якість
6	За інтенсивністю: марочна	Наявність такого зразка, який би позитивно відрізняв його від його прямих конкурентів	Впровадження унікальної назви, логотипу та корпоративного стилю

Далі необхідно виконати аналіз конкуренції за М. Портером (табл. 5.9).

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаро замінники
	GeekBrains, HTML Academy	Інші розробники аналогічних систем	Відсутні	Приватні навчальні заклади, а також курси навчання	Відсутні

Закінчення таблиці 5.9

Висновки	Потрібно забезпечити високий рівень надійності, швидкодії, а також зручності використання кінцевим користувачем	За допомогою розширення функціоналу системи є можливість стати конкурентоспроможним	Відсутні	Побажання щодо функціональних можливостей необхідно врахувати в процесі розробки персонального кабінету	Відсутні

Провівши аналіз отриманої інформації можна зробити висновок, що розроблений персональний кабінет є конкурентоспроможним, а також про існування конкуренції на ринку, але обидва розглянуті прямі конкуренти мають низку недоліків. Тому при запуску даного продукту необхідно розробляти з урахуванням усіх побажань користувачів, а також максимально обіграти явні недоліки аналогічних систем. Також проведено ступеневий аналіз конкуренції, визначено характеристики за якими вона проявляється, а також їх вплив на діяльність підприємства.

Наступним кроком необхідно обґрунтувати фактори конкурентоспроможності розробленого персонального кабінету.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Автоматизація процесу навчання	Зменшення часу необхідного для обліку навчальних груп, студентів та завантаження навчальних матеріалів за допомогою впровадження автоматизації навчального процесу
2	Висока швидкість роботи	Персональний кабінет працює із значно більшою швидкістю ні подібні системи конкурентів
3	Можливість масштабування системи	Даний додаток буде витримувати великі навантаження від великої кількості користувачів, які в один момент часу користуються системою

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін системи

№	Фактор конкуренто-спроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з новою системою						
			-3	-2	-1	0	+1	+2	+3
1	Автоматизація процесу навчання	10			+				
2	Висока швидкість роботи системи	15		+					
3	Можливість масштабування системи	10			+				

На основі аналізу отриманих даних сформуємо SWOT аналіз. На основі аналізу факторів загроз, а також можливостей маркетингового середовища необхідно скласти перелік ринкових можливостей та загроз.

Таблиця 5.12 – SWOT – аналіз стартап-проекту

Сильні сторони (S): • Висока швидкість роботи • зручність використання • можливість масштабування	Слабкі сторони(W): • коштовність розробки системи • вартість обслуговування
Можливості(O): • додання нових функціональних можливостей • підвищення швидкодії системи	Загрози(T): • відсутність фінансування • можлива поява конкурентів

За допомогою отриманого SWOT-аналізу необхідно створити альтернативи ринкової поведінки для того, що забезпечити запуску стартап проекту на ринку, а також оптимальний час ринкової реалізації.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання пробного терміну користування системою	Головний ресурс – люди, даний ресурс – наявний	1-3 місяці

Закінчення таблиці 5.13

2	Реклама	Використання особистих коштів для реклами системи	2-3 місяці
3	Публікація опису системи на відповідних ресурсах	Головний ресурс – час, даний ресурс – наявний	2-3 тижні
4	Презентація системи на відповідних публічних заходах	Ресурс – час та гроші для участі, наявні	1 місяць

За результатами проведеного аналізу можна зробити висновок, що необхідно обрати презентацію системи на відповідних публічних заходах для ринкового впровадження стартап-проекту.

5.4 Розробка ринкової стратегії стартап проекту

Визначення стратегії охоплення ринку передує саме розробленню ринкової стратегії стартап-проекту.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	IT спеціалісти. Вік: від 16 до 30. Місце проживання: не важливо. Сімейний стан: не важливий. Сфера зайнятості та рівень заробітної плати: IT сфера, від 15 тис. грн.	Даний персональний кабінет можна застосовувати для підвищення якості навчального процесу	Одна ліцензійна угода для фізичної або юридичної особи	За рахунок великого попиту на ринку на системи подібного типу, тому інтенсивність конкуренції висока	Сегмент дозволяє вийти на ринок та за рахунок його переваг над аналогічними системами показати його у гарному світлі

Закінчення таблиці 5.14

2	Приватні навчальні заклади. Сфера зайнятості – освіта.	Даний персональний кабінет вирішує питання підвищення якості та продуктивності навчального процесу.	Одна ліцензійна угода для фізичної або юридичної особи	За рахунок великого попиту на ринку на системи подібного типу, тому інтенсивність конкуренції висока	Сегмент дозволяє вийти на ринок та за рахунок його переваг над аналогічними системами показати його у гарному світлі
Які цільові групи обрано: ІТ сфера та приватні навчальні заклади, у яких виникає необхідність у підвищенні якості та продуктивності процесу навчання.					

Беручи до уваги той факт, що в ролі цільової групи визначені приватні навчальні заклади та в цілому освітня сфера, тому вирішено обрати стратегію масового маркетингу.

Таблиця 5.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання відсутніх функціональних можливостей, які відсутні у прямих аналогів даного додатку, розширена підтримка користувачів	Впровадження реклами, презентація демо версій та опис можливостей даного кабінету на відповідних ІТ конференціях, боротьба за прихильність користувачів до свого продукту	Зниження величини замінності товару його прямими аналогами; Прихильність заінтересованих користувачів; Позитивні властивості товару;	Диференціації

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, тому, що на ринку існують подібні системи схожого типу, але із значно меншими можливостями	Так, головна мета організації – це приплив нових користувачів, та/або часткове запозичення їх від аналогічних проектів	Дана організація копіює основні характеристик товару конкурентів, так-як вони є базовими для системи такого типу, але ставить за мету додання нових та унікальних функціональних можливостей	Заняття конкурентної ніші

Таблиця 5.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Відмінні характеристики веб-додатку	Стратегія диференціації	Унікальні функціональні можливості;	Безпека, Зручність, Унікальність
2	Технічна підтримка команди розробників	Стратегія диференціації	Технічна підтримка клієнтів; Оформлення ліцензії.	Турбота про клієнта
3	Слідування загальноприйнятим нормам проектування інтерфейсів	Стратегія диференціації	Вдосконалення загально прийнятих методів проектування інтерфейсів	Гнучкість, Стабільність

Після проведення аналізу, вирішено обрати в якості стратегії розвитку – стратегію диференціації та стратегію заняття конкурентної ніші.

5.5 Розробка маркетингової програми стартап-проекту

Таблиця 5.18 – Визначення переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Підвищення якості навчання приватного навчального закладу	Підвищення якості навчання приватних навчальних закладів за рахунок автоматизації учбового процесу	Існуючі конкуренти мають скромніші функціональні можливості та значно меншу швидкість роботи системи
2	Можливість навчання студентів відповідних спеціальностей	Студенти, які з певних причин відсутні на заняттях мають можливість переглянути навчальний матеріал онлайн	Існуючі конкуренти дають обмежений доступ основній групі студентів до вивчення навчальних матеріалів

Наступним кроком необхідно визначити опис рівнів моделі товару.

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Підвищення якості навчання у приватних навчальних закладах за рахунок автоматизації учбового процесу</td></tr>
<tr><td rowspan="7">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>Зручність використання</td><td>-</td><td>Висока</td></tr>
<tr><td>Забезпечення швидкої роботи та обробки даних</td><td>Час завантаження сторінок</td><td></td></tr>
<tr><td>Вартість</td><td>Грн.</td><td>0</td></tr>
<tr><td colspan="3">Якість: відповідність загальноприйнятим нормам</td></tr>
<tr><td colspan="3">Пакування: ліцензійна угода та сама система</td></tr>
<tr><td colspan="3">Марка: KursPC</td></tr>
<tr><td rowspan="2">3. Товар із підкріпленням</td><td colspan="3">До продажу: документація основних модулів системи, акційні пропозиції щодо придбання більше двох ліцензій</td></tr>
<tr><td colspan="3">Після продажу: надання кваліфікованої підтримки із технічних питань, а також додання додаткових можливостей</td></tr>
<tr><td colspan="4">Даний персональний кабінет буде захищено від копіювання за рахунок здійснення реєстрації власної марки програми, отримання патенту на програмний код даного додатку</td></tr>
</table>

Таблиця 5.20 – Визначення меж встановлення вартості

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
-	близько 40 тис. грн..	від 30 тис. грн.	20-30 тис. грн

Таблиця 5.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Отримання розробленого персонального кабінету	Функція ліцензування	Нульовий та/або однорівневий	Традиційна

Таблиця 5.22 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Отримання функціональних можливостей / системи	E-mail, телефон, форма зворотного зв’язку на сайті	Безпека та унікальність товару	Заохотити потенційних користувачів	Творча та класична
2	Отримання подальшої підтримки користувачів		Вчасна та кваліфікована підтримка користувачів	Запевнити користувача у наданні вчасної допомоги при виникненні питань технічного характеру	

На основі цього розроблено ринкову програму, яка складається із переліку ключових переваг концепції потенційного товару, опис його моделі, визначення питань ціно становлення а також питань формування системи збуту на основі концепції маркетингової комунікації.

Висновки до розділу

В даному розділі описано головну ідею стартап проекту. Наступним кроком проведено технологічний аудит ідеї проекту, в результаті якого визначено, що головна ідею проекту є здійсненною. На основі отриманих результатів проведено аналіз ринкових можливостей запуску стартап-проекту. Далі вирішено питання із розробкою ринкової стратегії проекту, а також із розробкою маркетингової програми стартап-проекту. Тому проаналізувавши всі пройдені етапи можна зробити висновок що даний проект може бути впроваджений на ринок, а також можна бути впевненим у зайнятті ним його відповідної ринкової ніші.

ВИСНОВКИ

В першому розділі зазначено головну мету даної магістерської дисертації, предмет та об’єкт наукового дослідження, а також вивчена область проведення цього дослідження. Наступним кроком проведено аналіз аналогічних існуючих систем. За аналогічні системи обрано систему дистанційного навчання GeekBrains, а також навчальні курси HTML Academy. Далі визначено їх переваги та недоліки, а також зазначено їх особливості реалізації. На основі отриманих даних поставлено задачу на дану магістерську дисертацію.

В другому розділі обрано необхідні технології для розробки персонального кабінету студента курсів навчання робототехніці. В процесі аналізу існуючих технологій обрано мову програмування PHP, зокрема його фреймворк Laravel в якості технології розробки серверної логіки системи, реляційну базу даних для зберігання користувацької інформації, а також мову програмування JavaScript, а саме її фреймворк React для розробки користувацького інтерфейсу.

В третьому розділі спроектовано архітектуру майбутнього персонального кабінету. Також спроектовано структуру бази даних та розроблено моделі для зручного її використання. Наступним кроком розроблено REST API для забезпечення взаємодії серверної частини та клієнтської. Далі розроблено основні модулі клієнтської та серверної частини. На завершаючому етапів проектування інтегровано систему кешування Redis, виконано тестування основних модулів серверної частини за допомогою unit-тестування, а також виконано масштабування серверу за допомогою Amazon EC2 для забезпечення безперебійного функціонування системи при пікових навантаженнях.

В наступному розділі проведено оптимізацію роботи персонального кабінету. За критерій оптимізації обрано швидкість завантаження сторінок. Проведено оптимізацію бази даних, а саме визначено прості запити які в

подальшому об’єднано в більш складні. Також модернізовано метод кешування індексів БД, а саме створення окремого додаткового кешу для тих індексів які використовуються у момент максимального навантаження системи, щоб вони завжди знаходились в оперативній пам’яті. Така реалізація по-перше позбавить сервер від завантаження кешу по запиту, тобто пришвидшить виконання перших запитів, а по-друге завантажений таким чином індекс буде повністю відсортований, що додатково пришвидшить виконання запитів із використанням такого кешу. Також проведена оптимізація серверної частини додатку, а саме виконано стиснення запитів, об’єднання декількох запитів в один та налаштування ресурсів веб-серверу. В останню чергу виконано оптимізацію клієнтської частини, а саме стиснення зображень, стиснення та об’єднання додаткових скриптів, а також використання кешу браузеру.

На завершення проведено маркетинговий аналіз стартап-проекту, в результаті якого визначено всі необхідні параметри та зроблено відповідні висновки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мова розмітки HTML [Електронний ресурс]: https://uk.wikipedia.org/wiki/HTML
2. Мова стилів CSS [Електронний ресурс]: https://uk.wikipedia.org/wiki/CSS
3. Мова програмування Javascript [Електронний ресурс]: https://uk.wikipedia.org/wiki/JavaScript
4. Документація мови програмування JavaScript [Електронний ресурс]: https://learn.javascript.ru/
5. Документація Bootstrap [Електронний ресурс] : http://getbootstrap.com/
6. Документація MySQL [Електронний ресурс] : https://www.mysql.com/
7. Дейв Энсор, Йен Стивенсон Oracle. Проектирование баз данных / Лорі, 2006. – 560 с.
8. Маркин А. В. Построение запросов и программирование на SQL / Питер Кому, 2008. – 704 с
9. Офіційна документація PHP [Електронний ресурс] : http://php.net/
10. Офіційна документація по фреймворку Laravel [Електронний ресурс] : https://laravel.com/
11. Документація по фреймворку Laravel [Електронний ресурс] : https://laravel.com/
12. Опис технології RESTfull [Електронний ресурс] : https://uk.wikipedia.org/wiki/REST

ДОДАТКИ

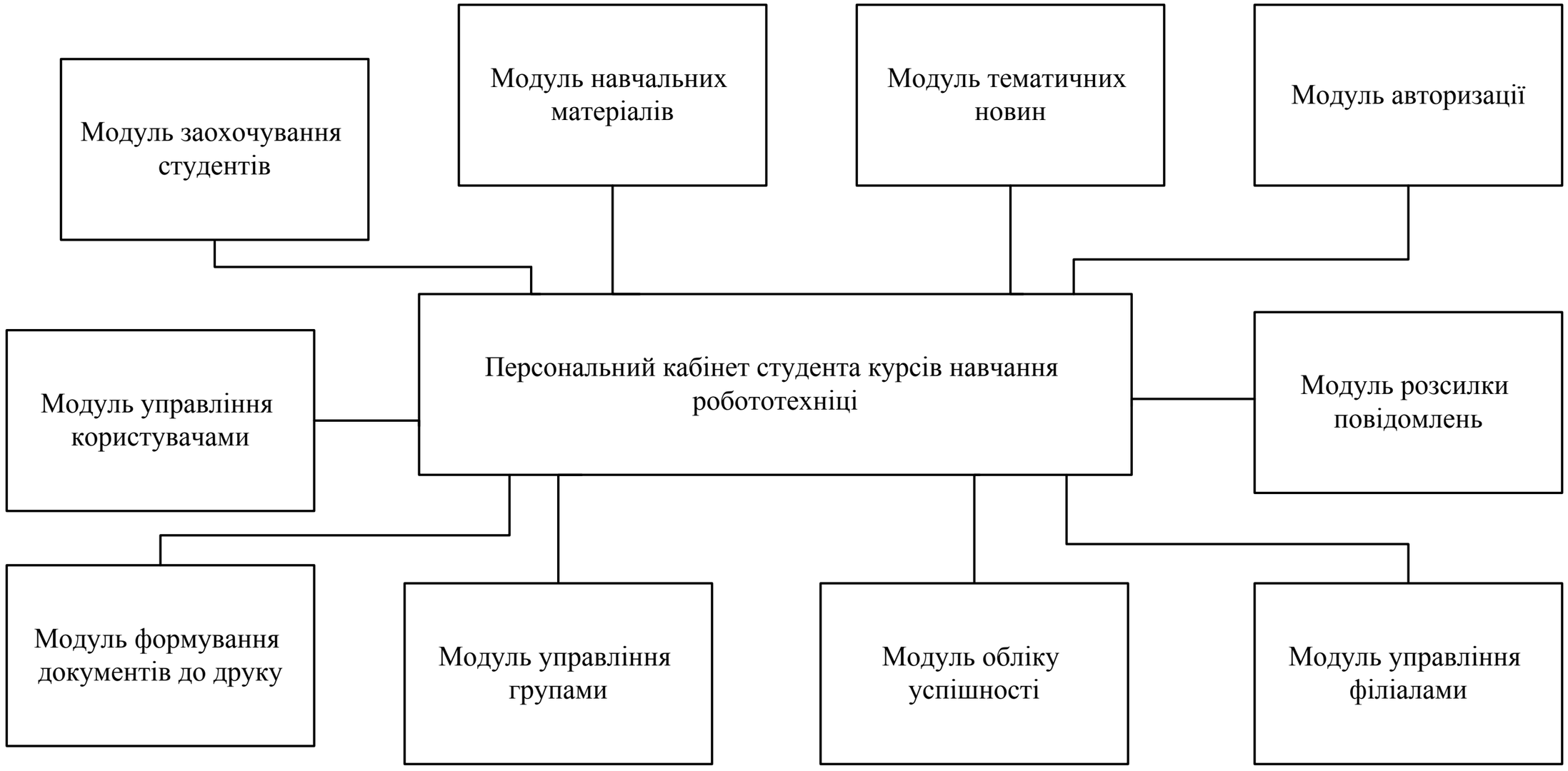
ДОДАТОК А

Плакати

ДОДАТОК Б

Результат перевірки на співпадіння

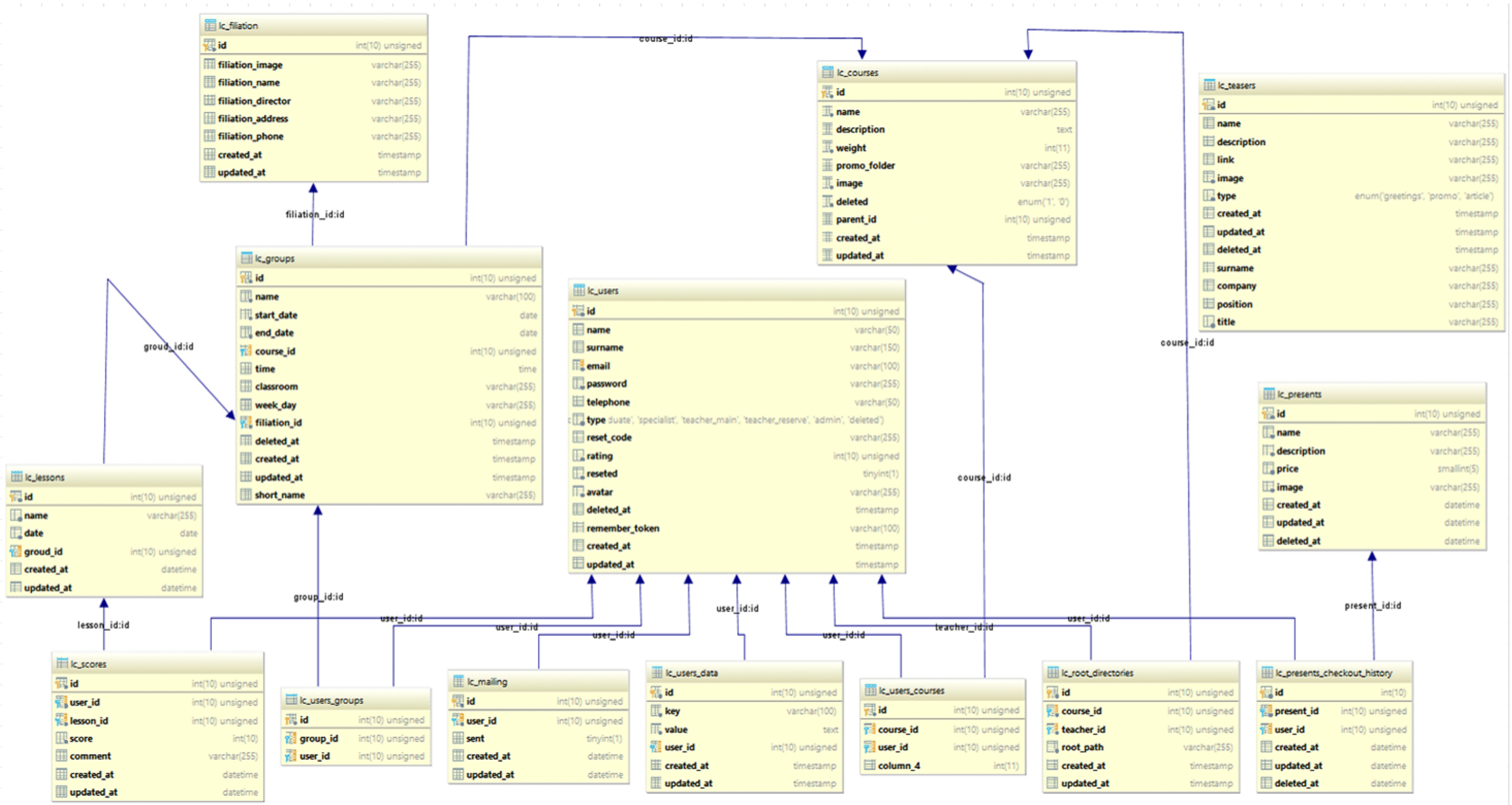
Структура персонального кабінету



Демонстраційний плакат №1
до магістерської дисертації на тему
«Персональний кабінет студента курсів навчання робототехніці»

Виконав: студент гр. ІК-82мп Новодранов А.С.
Керівник: к.т.н., доцент Крилов Є.В.

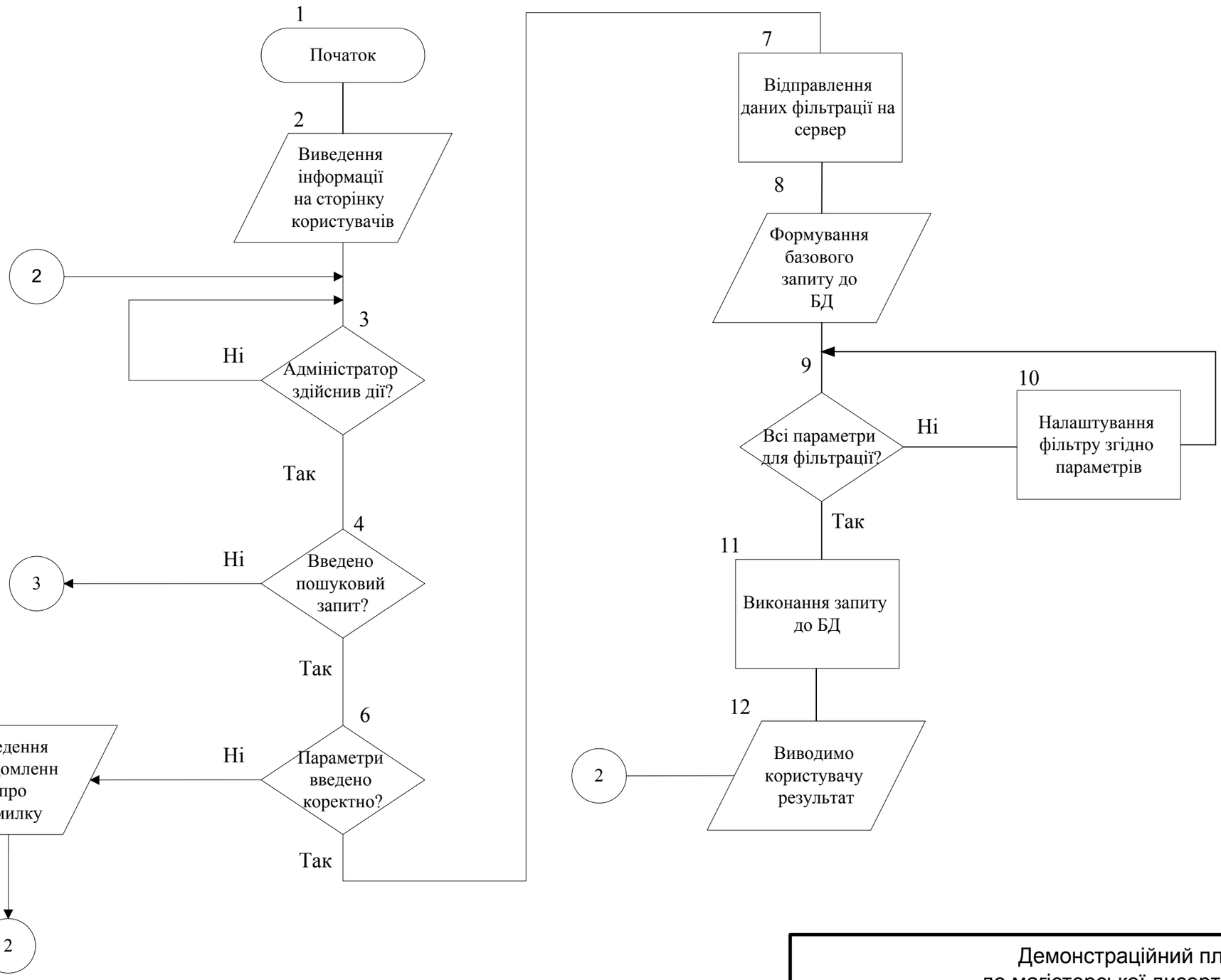
Структура бази даних системи



Демонстраційний плакат №2 до магістерської дисертації на тему «Персональний кабінет студента курсів навчання робототехніці»

Виконав: студент гр. ІК-82мп Новодранов А.С.
Керівник: к.т.н., доцент Крилов Є.В.

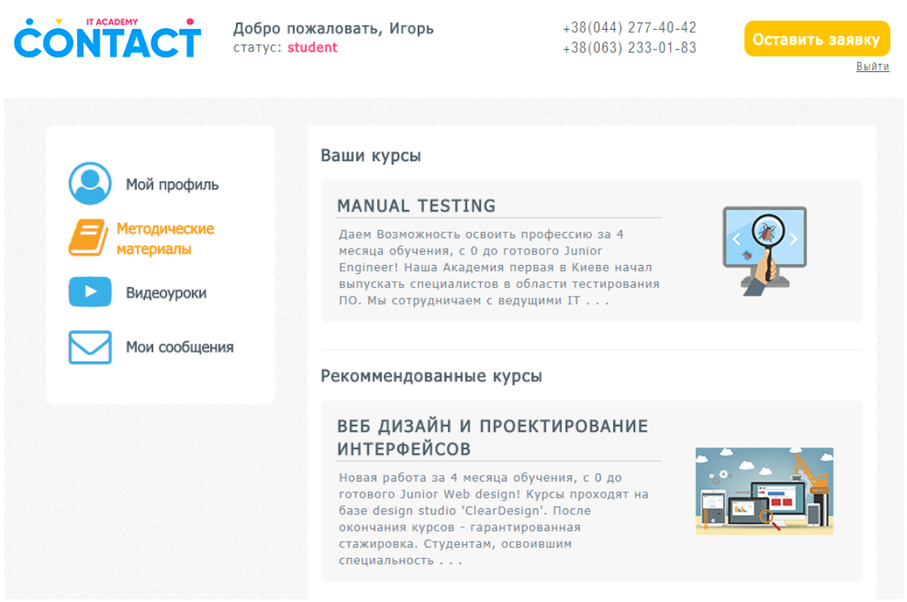
Алгоритм фільтрації користувачів



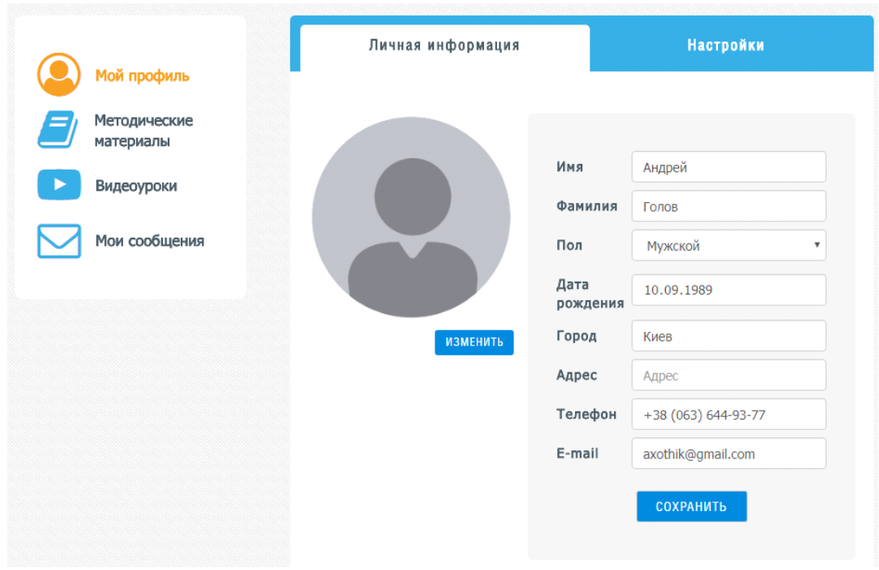
Демонстраційний плакат №3
до магістерської дисертації на тему
«Персональний кабінет студента курсів навчання робототехніці»

Виконав: студент гр. ІК-82мп Новодранов А.С.
Керівник: к.т.н., доцент Крилов Є.В.

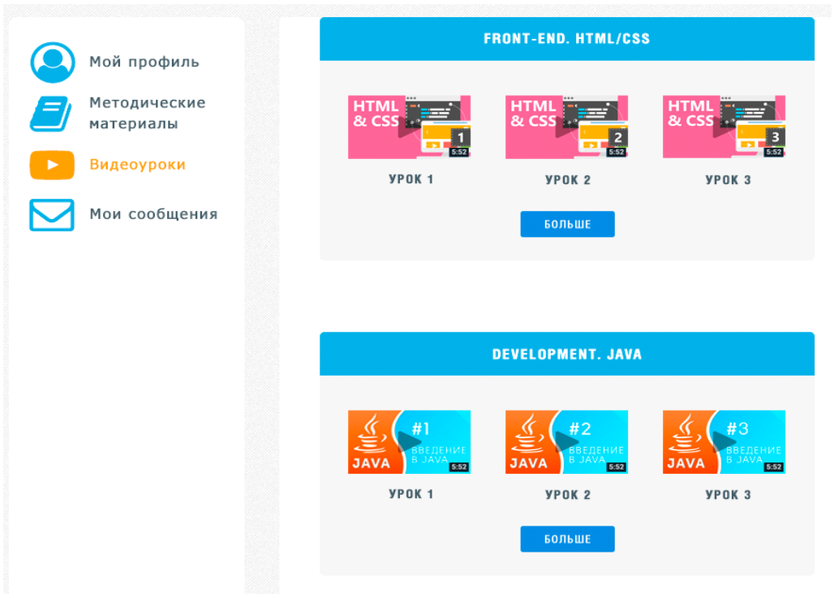
Інтерфейс користувача



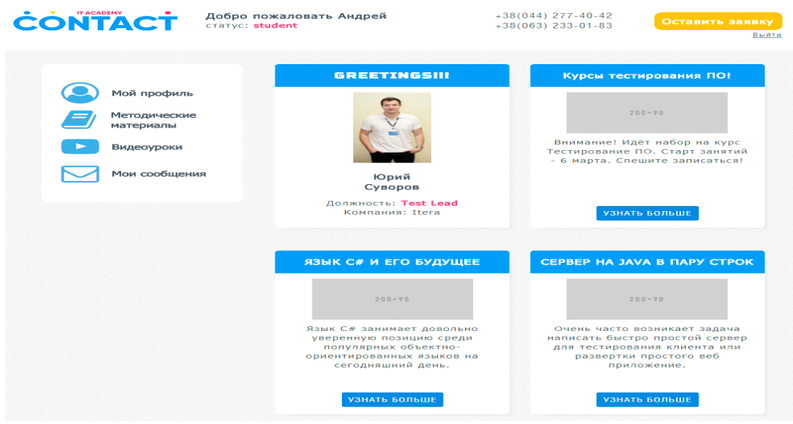
Сторінка навчальних курсів



Сторінка користувача



Сторінка відео матеріалів

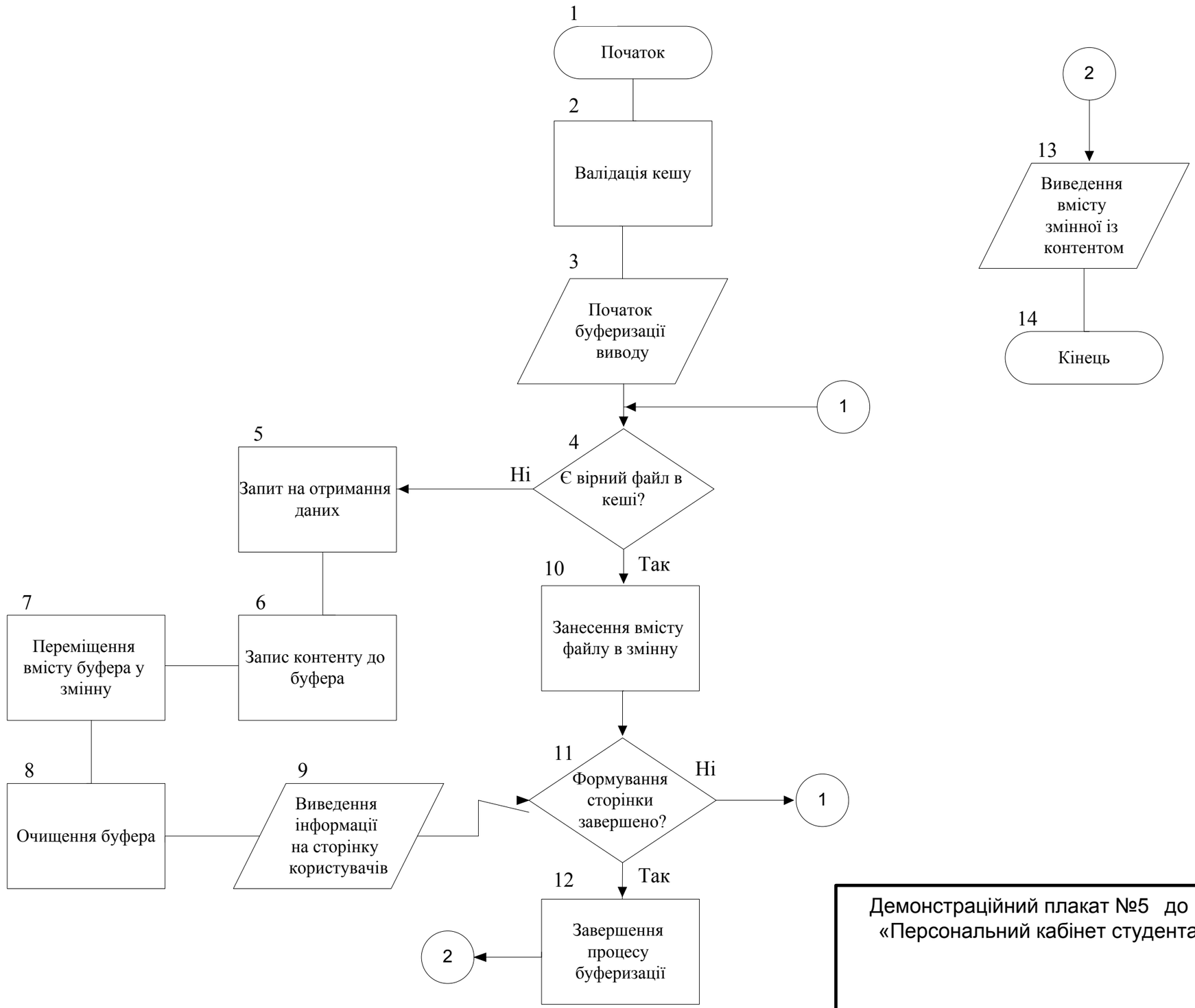


Головна сторінка персонального кабінету

Демонстраційний плакат №4
до магістерської дисертації на тему
«Персональний кабінет студента курсів навчання робототехніці»

Виконав: студент гр. ІК-82мп Новодранов А.С.
Керівник: к.т.н., доцент Крилов Є.В.

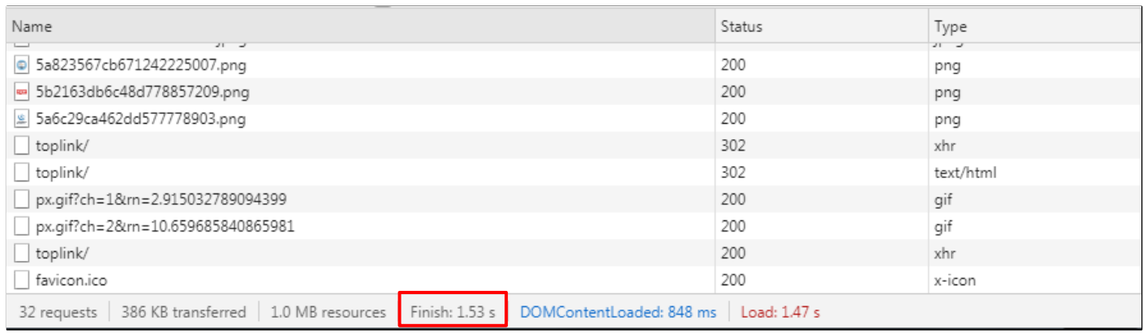
Алгоритм кешування додатку



Демонстраційний плакат №5 до магістерської дисертації на тему «Персональний кабінет студента курсів навчання робототехніці»

Виконав: студент гр. ІК-82мп Новодранов А.С.
Керівник: к.т.н., доцент Крилов Є.В.

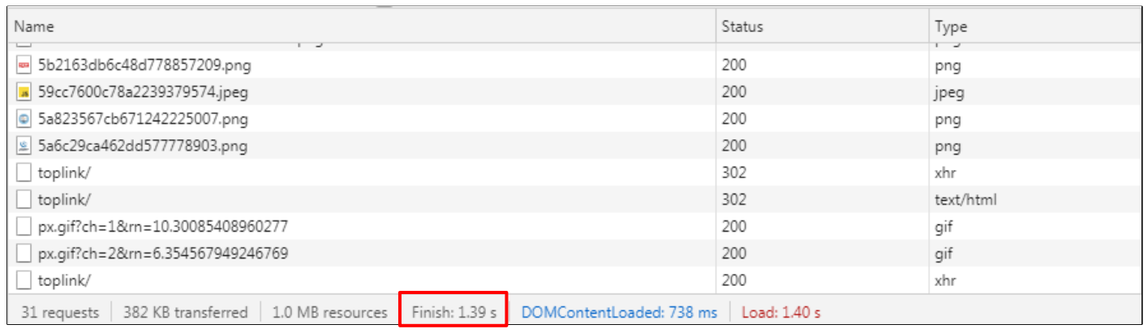
Швидкодія персонального кабінету



Name	Status	Type
5a823567cb671242225007.png	200	png
5b2163db6c48d778857209.png	200	png
5a6c29ca462dd577778903.png	200	png
toplink/	302	xhr
toplink/	302	text/html
px.gif?ch=1&rn=2.915032789094399	200	gif
px.gif?ch=2&rn=10.659685840865981	200	gif
toplink/	200	xhr
favicon.ico	200	x-icon

32 requests | 386 KB transferred | 1.0 MB resources | Finish: 1.53 s | DOMContentLoaded: 848 ms | Load: 1.47 s

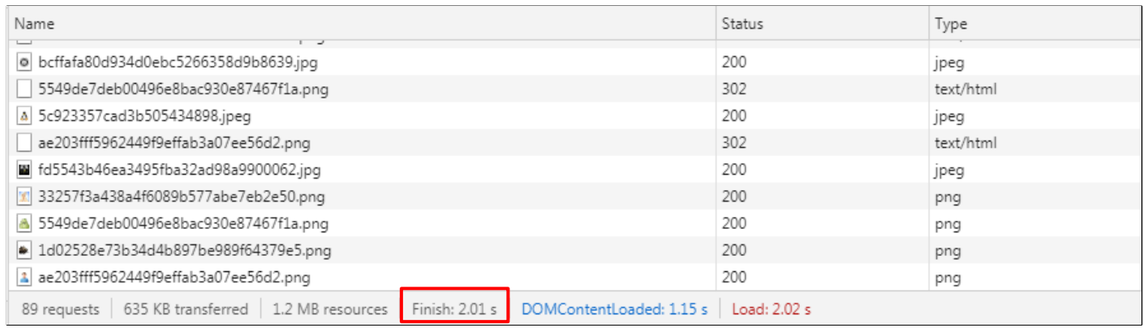
Завантаження головної сторінки персонального кабінету



Name	Status	Type
5b2163db6c48d778857209.png	200	png
59cc7600c78a2239379574.jpeg	200	jpeg
5a823567cb671242225007.png	200	png
5a6c29ca462dd577778903.png	200	png
toplink/	302	xhr
toplink/	302	text/html
px.gif?ch=1&rn=10.30085408960277	200	gif
px.gif?ch=2&rn=6.354567949246769	200	gif
toplink/	200	xhr

31 requests | 382 KB transferred | 1.0 MB resources | Finish: 1.39 s | DOMContentLoaded: 738 ms | Load: 1.40 s

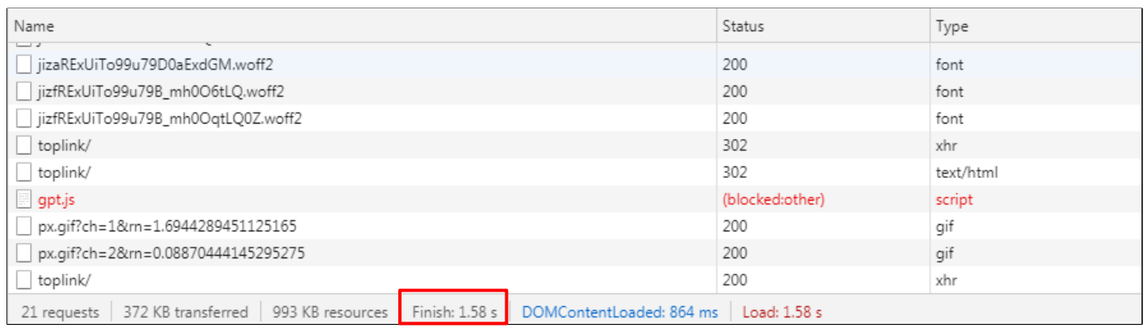
Завантаження сторінки навчальних курсів



Name	Status	Type
bcffafa80d934d0ebc5266358d9b8639.jpg	200	jpeg
5549de7deb00496e8bac930e87467f1a.png	302	text/html
5c923357cad3b505434898.jpeg	200	jpeg
ae203fff5962449f9effab3a07ee56d2.png	302	text/html
fd5543b46ea3495fba32ad98a9900062.jpg	200	jpeg
33257f3a438a4f6089b577abe7eb2e50.png	200	png
5549de7deb00496e8bac930e87467f1a.png	200	png
1d02528e73b34d4b897be989f64379e5.png	200	png
ae203fff5962449f9effab3a07ee56d2.png	200	png

89 requests | 635 KB transferred | 1.2 MB resources | Finish: 2.01 s | DOMContentLoaded: 1.15 s | Load: 2.02 s

Завантаження сторінки відповідного курсу



Name	Status	Type
jizaRExUiTo99u79D0aExdGM.woff2	200	font
jizfRExUiTo99u79B_mh0O6tLQ.woff2	200	font
jizfRExUiTo99u79B_mh0OqtLQ0Z.woff2	200	font
toplink/	302	xhr
toplink/	302	text/html
gpt.js	(blocked:other)	script
px.gif?ch=1&rn=1.6944289451125165	200	gif
px.gif?ch=2&rn=0.08870444145295275	200	gif
toplink/	200	xhr

21 requests | 372 KB transferred | 993 KB resources | Finish: 1.58 s | DOMContentLoaded: 864 ms | Load: 1.58 s

Завантаження сторінки користувача

Демонстраційний плакат №6 до магістерської дисертації на тему «Персональний кабінет студента курсів навчання робототехніці»

Виконав: студент гр. ІК-82мп Новодранов А.С.
Керівник: к.т.н., доцент Крилов Є.В.